CLARKSON UNIVERSITY

A Computational Approach to Measuring Homeomorphic Defect

A thesis

by

Scott M. LaLonde

Department of Mathematics and Computer Science

Submitted in partial fullfillment of the requirements

for the degree of

Master of Science

Mathematics

Date

Accepted by the Graduate School

Date

Dean

CLARKSON UNIVERSITY

The undersigned have examined the dissertation entitled **A Computational Approach to Measuring Homeomorphic Defect** presented by **Scott M. LaLonde** a candidate of the degree of **Master of Science** and hereby certify that it is worthy of acceptance.

Examining Committee

Erik Bollt (Advisor)	**Date**
Joseph Skufca (Advisor)	**Date**
Takashi Nishikawa	**Date**

A Computational Approach to Measuring Homeomorphic Defect

Abstract

A Computational Approach to Measuring Homeomorphic Defect

by

Scott M. LaLonde

Master of Science in Mathematics

Clarkson University

An important concept in the field of dynamical systems is the notion of conjugacy. Two dynamical systems are said to be conjugate if their dynamics are topologically equivalent. In other words, there is a homeomorphism between the underlying spaces which preserves the dynamics of the two systems. In this thesis we will be discussing an extension of this idea called mostly conjugacy.

In the context of mostly conjugacy, we deal with functions called commuters. These relate two dynamical systems that are not necessarily conjugate. We can determine the amount by which two dynamical systems fail to be conjugate by studying certain properties of their associated commuter. As commuters are not generally homeomorphisms, we will do so by studying a quantity called the homeomorphic defect. The work presented here is devoted largely to developing computational techniques for measuring this defect. In particular, we will construct an algorithm for approximating the Lebesgue measure of subsets of $\mathbb{R}^n$ which will rely heavily on the concept of Monte Carlo integration.

Once we have constructed the algorithm, we will present results from its deployment on various benchmark sets. We will analyze these results and use them to present arguments for the validity of the algorithm. Finally, we will discuss open questions and possible future work that can be done to achieve the goal of measuring homeomorphic defect to a reasonable degree of accuracy.

Contents

List of Figures

List of Tables

Acknowledgements

I would like to thank my advisor, Dr. Erik Bollt, and Dr. Joseph Skufca for their guidance throughout the course of my work on this topic. I would not be writing this were it not for their wisdom and their ability to always push me in the right direction.

I am also grateful to the rest of the faculty of the Division of Mathematics and Computer Science at Clarkson University for giving me much of the knowledge necessary to complete this work. Throughout both my undergraduate and graduate careers, they have given me a strong background in both pure and applied mathematics that has been critical to my achievements thus far.

I would also like to thank the committee members, Erik Bollt, Joe Skufca, and Takashi Nishikawa for investing their time in reading and commenting on this thesis.

Finally, it should be noted that this work was supported by the National Science Foundation through Grant No. DMS-0708083. Without this funding, this work would not have been possible.

Chapter 1

Background

1.1 Introduction

In this introductory chapter we will be discussing relevant background material for the study of conjugate dynamical systems. We will first discuss the concept of conjugacy and its overall importance to the field. We will then generalize this notion, leading to the definition of mostly conjugacy. Finally, we will discuss appropriate notation and assumptions for computing the measure of mostly conjugacy. This in turn will motivate the remainder of the work in the following two chapters. The majority of the material regarding conjugacy can be found in Alligood, Sauer, and Yorke [2], while further discussion of mostly conjugacy and related topics is located in Bollt and Skufca [12, 13].

1.2 Conjugacy

A well-known concept in the field of dynamical systems is the notion of conjugacy. Intuitively, we could say that conjugate dynamical systems are equivalent in a certain sense. More specifically, conjugacy defines an equivalence relation on the set of all dynamical systems. If two dynamical systems are conjugate, then they generate precisely the same dynamics. This somewhat vague description can be formalized by the following definition.

Definition 1.1. *Let X and Y be topological spaces, and let $g_1 : X \to X$ and $g_2 : Y \to Y$. The dynamical systems g_1 and g_2 are* ***conjugate*** *if there exists a homeomorphism $h : X \to Y$*

such that

$$g_1(x) = (h^{-1} \circ g_2 \circ h)(x) \tag{1.1}$$

for all $x \in X$.

Since h is a homeomorphism, it is a continuous bijection with a continuous inverse. It follows that h serves as a change of coordinates between g_1 and g_2. Also, to formally restate an idea that was mentioned earlier, we can say that the dynamics generated by g_1 and g_2 are equivalent in a topological sense.

The importance of conjugacy lies in the fact that if g_1 and g_2 are conjugate, they generate the same dynamics. Because of this, we can study the dynamics of g_1 by either analyzing g_1 directly or by instead working with g_2. This is particularly useful if one dynamical system is much easier to handle computationally than the other. For example, if g_1 is particularly complicated, but g_2 is not, we can choose to study g_1 by instead evaluating g_2.

The following example illustrates a well-known example of a pair of conjugate dynamical systems, the full-shift tent map and the logistic map. The analysis here is based on a similar explanation found in [2].

Example 1.2. Let g_1 be the symmetric tent map of height 1,

$$g_1(x) = \begin{cases} 2x & \text{if } 0 \le x \le 1/2 \\ 2(1-x) & \text{if } 1/2 < x \le 1 \end{cases}$$

and let g_2 be the logistic map with parameter 4,

$$g_2(x) = 4x(1-x).$$

Figure 1.2 shows the graphs of these two functions. We can see that both functions map the interval $[0,1]$ to $[0,1]$. These dynamical systems are related by the conjugacy $h : [0,1] \to [0,1]$ defined by

$$h(x) = \frac{1-\cos(\pi x)}{2}.$$

This function is shown in Figure 1.2.

We can easily verify that h is a homeomorphism. It is clearly continuous, and

$$h'(x) = \frac{\pi}{2}\sin(\pi x),$$

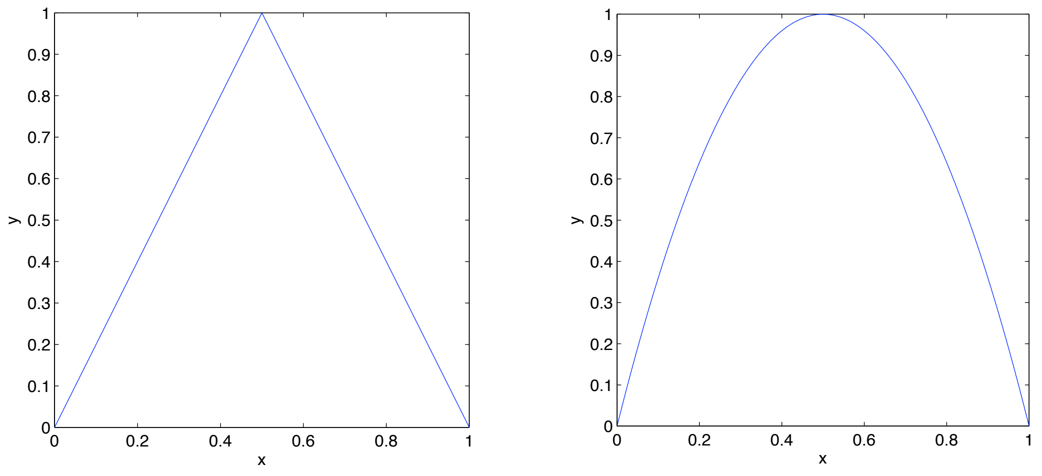


Figure 1.1. The tent map g_1 (left) and the logistic map g_2 (right).

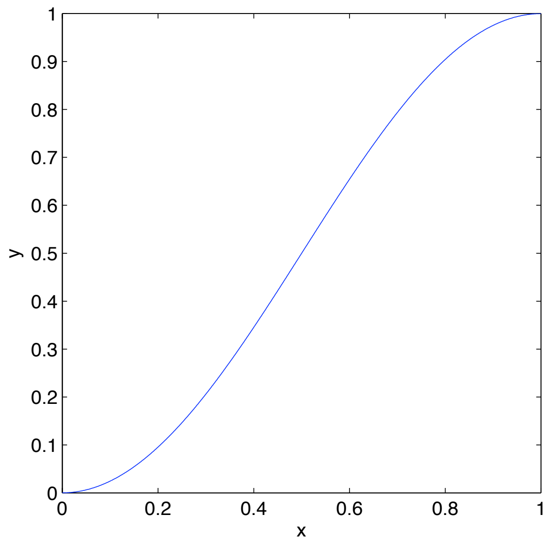


Figure 1.2. The conjugacy $h(x)$.

which is strictly positive on $(0,1)$. Thus h is increasing on $[0,1]$, hence it is one-to-one. Also, $h(0)=0$ and $h(1)=1$, so by continuity (specifically, the intermediate value theorem) h maps $[0,1]$ onto itself. Finally, the inverse of h is given by

$$h^{-1}(x)=\frac{1}{\pi}\cos^{-1}(1-2x),$$

which is continuous, as h is continuous and one-to-one on $[0,1]$. Thus h is a homeomorphism.

We will omit the verification that h satisfies the equation $g_1=h^{-1}\circ g_2\circ h$, as such a calculation can be found in [2]. However, we will mention that the procedure can be made simpler by rewriting the equation in the form $h\circ g_1=g_2\circ h$ and verifying that the two sides are equal. The idea of writing the equation this way will quickly prove useful and it is of paramount importance to the work introduced in the next section.

1.3 Mostly Conjugacy

In Example 1.2 we mentioned that (1.1) can be written in the equivalent form

$$(h\circ g_1)(x)=(g_2\circ h)(x) \tag{1.2}$$

in order to verify the conjugacy of the tent map and the logistic map. Let us note that verification that h satisfies this expression does not require knowledge of h^{-1}; in fact, if we simply consider (1.2), by itself, there is no indication that h even has a well-defined inverse. Naturally, this leads us to ask whether it may be possible to relax certain conditions in the definition of conjugacy in an effort to study the relationship between two arbitrary dynamical systems. It is in this context that the idea of mostly conjugacy naturally arises.

We would like to know if we can, in general, find a function f which satisfies the equation

$$(f\circ g_1)(x)=(g_2\circ f)(x). \tag{1.3}$$

This expression is referred to as the commuting relationship. With regard to the function f, the following piece of terminology will prove useful throughout the remainder of this thesis.

Definition 1.3. *Let X and Y be topological spaces, and let $g_1:X\to X$ and $g_2:Y\to Y$. If $f:X\to Y$ satisfies the commuting relationship (1.3), then we say f is a* ***commuter*** *relating the dynamical systems g_1 and g_2.*

Note that unlike in Definition 1.1, we impose no restrictions on the properties of the function f. In particular, f may not necessarily be a homeomorphism. We simply define commuters for arbitrary pairs of dynamical systems, regardless of whether the systems are conjugate or not.

There are two important questions which arise naturally regarding commuters. The first of these regards the existence and computability of commuters. That is, do commuters actually exist in general, and, if so, is it possible to construct them on a computer? Generally, the answer to each part of this question is yes. We present the following simple example, which shows how to construct the commuter for two non-conjugate tent maps. Commuters of this nature will appear repeatedly throughout the remainder of this work.

Example 1.4. Let g_2 denote the standard tent map that was mentioned in Example 1.2, and let g_1 be the symmetric tent map of height 3/4,

$$g_1(x) = \begin{cases} \frac{3}{2}x & \text{if } 0 \leq x \leq \frac{1}{2} \\ \frac{3}{2}(1-x) & \text{if } \frac{1}{2} < x \leq 1. \end{cases}$$

We would like to find a function f which satisfies the commuting relationship for these two maps.

First, consider the case when $x \in [0, 1/2]$. The left side of (1.3) becomes

$$f(g_1(x)) = f\left(\frac{3}{2}x\right).$$

To determine the right side, we must evaluate $g_2(f(x))$. This in turn depends on where $f(x)$ falls for $x \in [0, 1/2]$. It is not entirely apparent whether $f(x) \in [0, 1/2]$ or $f(x) \in (1/2, 1]$, thus it is not clear which branch of g_2 should be used. This dilemma can be solved in the following way. We will require that the commuter f maps monotone segments of g_1 to monotone segments of g_2. That is, if g_1 is monotone increasing (respectively, decreasing) on the interval $[a, b]$, then g_2 is also monotone increasing (respectively, decreasing) on $f([a, b])$. Thus we know that if $x \in [0, 1/2]$, $f(x) \in [0, 1/2]$, so

$$g_2(f(x)) = 2f(x).$$

Setting this equal to the previous result and solving for $f(x)$ gives

$$f(x) = \frac{1}{2}f\left(\frac{3}{2}x\right).$$

If $x \in (1/2, 1]$, we have

$$f(g_1(x)) = f\left(\frac{3}{2}(1-x)\right)$$

and

$$g_2(f(x)) = 2(1 - f(x)),$$

so

$$f(x) = 1 - \frac{1}{2} f\left(\frac{3}{2}(1-x)\right).$$

Thus the commuter f can be expressed as

$$f(x) = \begin{cases} \frac{1}{2} f\left(\frac{3}{2}x\right) & \text{if } 0 \le x \le \frac{1}{2} \\ 1 - \frac{1}{2} f\left(\frac{3}{2}(1-x)\right) & \text{if } \frac{1}{2} < x \le 1. \end{cases}$$

Note that this is a functional equation - it is defined implicitly in terms of f. While we do not have a closed-form expression for f, we can in practice use an iterative process for computing an approximation to the function. This process is described in more detail in [13] and [12]. Figure 1.4 shows a graph of f generated in this way.

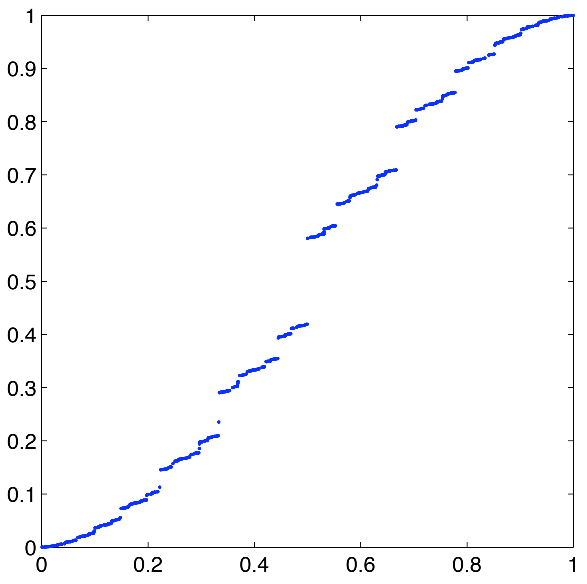


Figure 1.3. Graph of commuter f, evaluated at 1000 evenly spaced points on $[0, 1]$.

In general, the derivation of a commuter is not as simple as the one we have just seen. However, there is an iterative procedure outlined in [13] which allows for the computational construction of a commuter in a more general setting. This procedure has been shown to be effective in most practical applications. Throughout this thesis, however, we will primarily be dealing with commuters that can be constructed in the manner of Example 1.4.

Now that we have addressed the existence of commuter functions, there is a second, equally important question which logically follows. In the event that g_1 and g_2 are not conjugate, can we somehow quantify the amount by which f fails to be a conjugacy? Also, even if we can theoretically define the distance from conjugacy, how can we go about measuring such a thing in practice? We will begin to address this issue in the following section, where background material relating to this problem will be introduced. The rest of this thesis will be devoted to developing computational techniques for measuring certain properties of f, which should in turn lead to techniques for determining how far f is from being a conjugacy.

1.4 Measure of Mostly Conjugate

Given a function f which satisfies the commuting relationship (1.3), we would like to determine how close f is to being a conjugacy. This in turn will tell us how close the dynamical systems g_1 and g_2 are to being conjugate. Let us note that g_1 and g_2 are conjugate if and only if f is a homeomorphism. Thus it would seem reasonable to measure the distance from conjugacy by measuring how close f is to being a homeomorphism. We will do so by calculating a quantity called the **homeomorphic defect** of f.

The homeomorphic defect of f, which we will denote by $\lambda(f)$, was introduced in [13]. Recall that f must satisfy four distinct properties in order to be a homeomorphism, namely it must be continuous, one-to-one, onto, and its inverse must also be continuous. As a result, it seems natural that $\lambda(f)$ should consist of four corresponding components. These are:

- $\lambda_O(f)$, the amount by which f fails to be onto;
- $\lambda_{1-1}(f)$, the amount by which f fails to be one-to-one;
- $\lambda_C(f)$, the amount by which f fails to be continuous;
- $\lambda_{C^{-1}}(f)$, the amount by which f^{-1} fails to be continuous.

We will explore the actual definitions of these defect measures as we progress. Let us note that each defect is nonnegative, and each vanishes if f satisfies the particular property (the converse, however, is not necessarily true). For example, $\lambda_O(f) \geq 0$, with equality if f is onto. The homeomorphic defect is a linear combination of the individual defects,

$$\lambda(f) = \alpha_1 \lambda_O(f) + \alpha_2 \lambda_{1-1}(f) + \alpha_3 \lambda_C(f) + \alpha_4 \lambda_{C^{-1}}(f), \tag{1.4}$$

where we require that $0 \leq \alpha_i \leq 1$ for each i and $\sum \alpha_i = 1$. The ambiguity in the definition of the weights α_i is intentional, as it allows the measure to be properly tuned in order to suit a particular application.

The rest of this thesis will be devoted to developing computational techniques for calculating the four defect measures described above. In particular, we will be focusing on the onto defect λ_O, since it has the simplest and most intuitive definition. Additionally, the bulk of the work that will be done to allow the onto defect to be measured will be necessary in order to measure the other components of the homeomorphic defect. In order to properly discuss this work and the associated mathematical theory, we must first introduce some notation that will be used heavily in the following chapter.

1.4.1 Notation and Assumptions

Thus far, we have been working under the assumption that the sets X and Y are simply arbitrary topological spaces. However, we will be defining the components of the homeomorphic defect using concepts from measure theory. As a result, we will need to assume that X and Y, or at least certain subsets of them, are endowed with a somewhat richer structure.

Let $D_1 \subset X$ and $D_2 \subset Y$. These sets are the subsets of X and Y which are important to the modeler; that is, D_1 and D_2 are chosen based on their significance to a particular application. For example, D_1 and D_2 might be taken to be the forward invariant sets under g_1 and g_2, respectively. In most practical applications, and in all examples detailed in this thesis, we will assume that $D_1, D_2 \subset \mathbb{R}^n$ with the relative topology inherited from the standard topology on $\mathbb{R}^n$.

Given D_1 and D_2, we also require that there are appropriate measures defined on these sets. That is, we assume that (D_1, Σ_1, μ_1) and (D_2, Σ_2, μ_2) are measure spaces. Furthermore, we assume that μ_1 and μ_2 are both finite and nonatomic[1] measures. It should also

[1]By *nonatomic*, we mean that μ_1 and μ_2 do not concentrate measure on finite or countable sets.

be noted that μ_1 and μ_2 are not necessarily the standard Lebesgue measure on $\mathbb{R}^n$. For example, the set D_1 in which we are interested may be a Cantor-like set with Lebesgue measure zero. It would be advantageous if we could choose μ_1 such that $\mu_1(D_1) > 0$, as this would allow us to meaningfully compare the sizes of subsets of D_1 to the size of the whole set. In this regard, the choice of μ_1 and μ_2 will also depend heavily on the particular problem to which these techniques are being applied.

Finally, there is one last piece of notation that we will introduce for future reference. In the interest of brevity and ease of notation, for a set $A \subset X$ we define

$$\bar{\mu}_2\left(f[A]\right) = \mu_2\left(f[A \cap D_1] \cap D_2\right). \tag{1.5}$$

Intuitively, this is a "restriction" of μ_2 to D_1 and D_2. More precisely, $\bar{\mu}_2$ measures the set of points in $f[A] \cap D_2$ which have preimages lying in D_1. Similarly, for $B \subset Y$ we define

$$\bar{\mu}_1\left(f^{-1}[B]\right) = \mu_1\left(f^{-1}[B \cap D_2] \cap D_1\right).$$

Both definitions will allow for more concise representations of the defect measures in the future.

1.5 Outline

We will conclude this introductory chapter with a brief summary of the remainder of this thesis. Now that we have discussed the relevant background material, Chapter 2 will be devoted to developing techniques for measuring the onto defect of a commuter f. This will require a brief introduction to Monte Carlo integration, which will be used to construct an algorithm for measuring subsets of $\mathbb{R}^n$ computationally. In Chapter 3, we will present results that have come from testing the algorithm on a collection of benchmark sets. We will provide a discussion of these results, as well as a heuristic analysis of the observed phenomena. Finally, Chapter 4 will provide a concluding discussion, as well as an outline of possible future work.

Chapter 2

Measurement of Onto Defect

2.1 Preliminaries

This chapter will be devoted to developing computational techniques for accurately measuring subsets of $\mathbb{R}^n$. This is done with the hope that it will ultimately lead to a method for calculating the onto defect of a specified commuter f. That is, we hope to compute a measure of, in some sense, how far f is from being a surjection. This is clearly a subjective measure, depending heavily on the subsets of the domain and target that we wish to consider. For example, a function clearly maps its domain onto its range, but we may be interested in a larger (or perhaps smaller) subset of the target. In this regard, we must be careful to specify the particular sets that are important to us.

Retaining the notation we introduced in Chapter 1, we will denote the sets of interest as D_1 and D_2. If $f : D_1 \to D_2$, a natural way to determine the onto defect would be to measure the fraction of D_2 which is not covered by the image of D_1 under f. This notion is formalized by the following definition, which is introduced by Bollt and Skufca [13].

Definition 2.1. *Let X and Y be topological spaces, and suppose (D_1, Σ_1, μ_1) and (D_2, Σ_2, μ_2) are measure spaces, with $D_1 \subset X$ and $D_2 \subset Y$. Let $f\colon D_1 \to D_2$. We define the* ***onto defect*** *of f to be*

$$\lambda_O(f) = 1 - \frac{\bar{\mu}_2(f[D_1])}{\mu_2(D_2)}, \tag{2.1}$$

where $\bar{\mu}_2$ is defined by (1.5).

Observe that this is simply a theoretical definition; for it to be of practical use, we must determine a technique for implementing it computationally.

If we are to have any hope of evaluating (2.1) on a computer, we must be able to reliably approximate the μ_2-measures of $f[D_1]$ and D_2. This will likely be a daunting task, since in most interesting cases these sets exhibit structures which are highly complicated. The situation appears more hopeful if we interpret the μ_2-measure of a set as an integral over that set. For example, given the set D_2 we can express $\mu_2(D_2)$ as

$$\mu_2(D_2) = \int_{D_2} 1 \, d\mu_2.$$

However, problems still remain. Were these sets in any way simple, we could apply classical numerical quadrature techniques to evaluate the associated integrals. Unfortunately, we do not have this luxury. In addition, the need to accommodate abstract measures further complicates the situation.

There is one numerical integration technique which may allow us to circumvent some of these issues. We can use a probabilistic scheme known as Monte Carlo integration to evaluate the integrals. This method allows for evaluation of integrals over arbitrary measurable sets in $\mathbb{R}^n$. It eliminates any need for explicit definitions of such sets, as it only requires the evaluation of their characteristic functions (or some suitable approximations to these functions). Also, if we make certain assumptions about the underlying problem, we can integrate with respect to measures other than the usual Lebesgue measure.

The bulk of this chapter will be spent developing methods for applying Monte Carlo integration to the problem of measuring the onto defect of a commuter f. We will first give a brief discussion of the method itself, including relevant background information and theoretical results. We will then consider specific issues that must be dealt with in order to make Monte Carlo integration a viable option for our problem. Computational considerations will then be discussed, including the statement of an algorithm that implements the theoretical ideas presented in the chapter. This will eventually lead to a discussion in Chapter 3 of numerical results coming from actual implementations of the algorithm.

2.2 Introduction to Monte Carlo Integration

The methods we will discuss later in this chapter depend heavily on knowledge of Monte Carlo integration, so we will begin by offering a brief introduction to the topic. We will explore its use for computing integrals in a general setting in $\mathbb{R}^n$. This is not meant to be an authoritative account of the method but simply a summarized adaptation of relevant results mentioned in Hammersley and Hanscomb [5] and Weinzierl [15]. The reader should consult those texts, or similar ones, for a more detailed discussion of the Monte Carlo method.

As mentioned earlier, Monte Carlo integration uses probabilistic methods to estimate the integral of a function over an arbitrary measurable subset of $\mathbb{R}^n$. This is accomplished in part by changing the region of integration to a simpler set, such as an interval. The function is evaluated at a discrete set of points selected from the interval, and these values are used to compute an approximation to the integral. This is a very rough explanation of the method; a more detailed and rigorous analysis follows.

2.2.1 Construction

Let E be a Lebesgue-measurable subset of $\mathbb{R}^n$ with finite measure and let $f \in L(E)$, where $L(E)$ denotes the set of all real-valued, Lebesgue-integrable functions on E. Suppose we wish to compute

$$\int_E f(x)\,dx, \tag{2.2}$$

where this is naturally understood to be a Lebesgue integral. Since E is an arbitrary measurable set, it may be difficult to develop a general procedure for approximating this integral numerically. It would be quite advantageous if we could somehow change the region of integration to a more elementary set. In this regard, the following result will prove useful.

Proposition 2.2. *Let $I \subset \mathbb{R}^n$ be an interval containing E, and let χ_E denote the characteristic function of E. Then*

$$\int_E f(x)\,dx = \int_I f(x)\chi_E(x)\,dx. \tag{2.3}$$

Proof. By definition, $\chi_E(x) = 1$ for all $x \in E$, so we may rewrite the integral in Eq. (2.2) as

$$\int_E f(x)\chi_E(x)\,dx$$

without changing its value. Now, since E and $I \setminus E$ are disjoint, we have

$$\int_I f(x)\chi_E(x)\,dx = \int_E f(x)\chi_E(x)\,dx + \int_{I\setminus E} f(x)\chi_E(x)\,dx.$$

But $\chi_E(x) = 0$ for all $x \in I \setminus E$, so the second integral vanishes, leaving

$$\int_I f(x)\chi_E(x)\,dx = \int_E f(x)\chi_E(x)\,dx.$$

We have already shown that the right side is equivalent to $\int_E f(x)\,dx$, so the proof is complete. $\square$

The result of Proposition 2.2 allows us to treat (2.2) as an integral over an interval $I \supseteq E$. That is, we now hope to approximate the quantity

$$\theta = \int_I f(x)\chi_E(x)\,dx. \tag{2.4}$$

The fact that we are now working on an interval in $\mathbb{R}^n$ is very useful computationally, since we can easily generate random numbers from a uniform distribution on I. This is of the utmost importance, since random sampling is critical to further development of this method.

We are now prepared to construct a numerical approximation to θ via probabilistic methods. Let $\{X_1, X_2, \ldots, X_N\}$ be an independent and identically distributed random sample taken from a uniform distribution on I. If we apply the function $f \cdot \chi_E$ to each of these random variables, we obtain a new set of independent and identically distributed random variables,

$$\{f(X_1)\chi_E(X_1), f(X_2)\chi_E(X_2), \ldots, f(X_N)\chi_E(X_N)\}. \tag{2.5}$$

The expected value of each of these random variables can be found by integrating over the entire sample space with respect to the uniform probability measure on I. This gives

$$\mathcal{E} = \frac{1}{|I|}\int_I f(x)\chi_E(x)\,dx. \tag{2.6}$$

Observe that the integral in (2.6) is precisely θ, which we are trying to approximate. Consequently, $\theta = \mathcal{E}|I|$. Since $|I|$ is easily computed, a statistical estimate of the expectation $\mathcal{E}$ will automatically provide an approximation to θ.

An appropriate estimator of $\mathcal{E}$ will be given by the sample mean of (2.5), which is

$$\bar{\mathcal{E}}_N = \frac{1}{N}\sum_{i=1}^{N} f(X_i)\chi_E(X_i).$$

By the law of large numbers (Theorem 4.8.4 of DeGroot and Schervish [3]),

$$\bar{\mathcal{E}}_N \xrightarrow{p} \mathcal{E},$$

where $\xrightarrow{p}$ denotes convergence in probability. Equivalently, given any $\varepsilon > 0$,

$$\lim_{N\to\infty} \Pr(|\bar{\mathcal{E}}_N - \mathcal{E}| < \varepsilon) = 1.$$

Thus if we define an approximation to θ by

$$\bar{\theta}_N = \frac{|I|}{N} \sum_{i=1}^{N} f(X_i)\chi_E(X_i), \tag{2.7}$$

we can expect that, for sufficiently large N, there is a high probability that $\bar{\theta}_N$ will be arbitrarily close to the true value of θ. This idea will be formalized in the next section. The expression in (2.7) is precisely the formula that will be used in practice to estimate the value of an integral of the form (2.2).

2.2.2 Error Analysis

In Section 2.2.1 we stated that the approximation $\bar{\theta}_N$ to θ will converge in probability to the true value of the integral as $N \to \infty$. We are now prepared to show this rigorously. To do so, we will need the following well-known inequality, taken from [3].

Theorem 2.3 (Tchebyshev Inequality). *Let X be a random variable for which $Var(X)$ exists. Then for every number $t > 0$,*

$$\Pr(|X - E[X]| \geq t) \leq \frac{Var(X)}{t^2}.$$

Note that this gives a probabilistic bound for the difference between a random variable and its expected value. In particular, the Tchebyshev inequality allows us to make the following statement regarding the difference between a sample mean and the true expectation of the associated random variables.

Lemma. *Let $X_1, X_2, \ldots, X_N$ be a random sample of size N taken from a distribution with mean μ and variance σ^2, and let $\bar{X}_N$ denote the sample mean. Then for every number $t > 0$,*

$$\Pr(\left|\bar{X}_N - \mu\right| \geq t) \leq \frac{\sigma^2}{Nt^2}.$$

Proof. By definition,

$$\bar{X}_N = \frac{1}{N}\left(X_1 + X_2 + \cdots + X_N\right).$$

This has expectation

$$\begin{aligned}\mathrm{E}[\bar{X}_N] &= \frac{1}{N}\left(\mathrm{E}[X_1] + \mathrm{E}[X_2] + \cdots + \mathrm{E}[X_N]\right) \\ &= \frac{1}{N}(N\mu) = \mu\end{aligned}$$

and variance

$$\begin{aligned}\mathrm{Var}[\bar{X}_N] &= \mathrm{Var}\left(\frac{1}{N}\left(X_1 + X_2 + \cdots + X_N\right)\right) \\ &= \frac{1}{N^2}\left(\mathrm{Var}(X_1) + \mathrm{Var}(X_2) + \cdots + \mathrm{Var}(X_N)\right) \\ &= \frac{1}{N^2}\left(N\sigma^2\right) = \frac{\sigma^2}{N},\end{aligned}$$

where we have used the fact that $X_1, X_2, \ldots, X_N$ are independent. Now, applying Tchebyshev's inequality, we have

$$\begin{aligned}\Pr(\left|\bar{X}_N - \mu\right| \geq t) &= \Pr(\left|\bar{X}_N - \mathrm{E}[\bar{X}_N]\right| \geq t) \\ &\leq \frac{\mathrm{Var}(\bar{X}_N)}{t^2} \\ &= \frac{\sigma^2}{Nt^2}\end{aligned}$$

for all $t > 0$. □

Given this result, we can now prove the following claim regarding the convergence of the Monte Carlo integration algorithm. The proof will also give us an estimate of the associated error.

Theorem 2.4. *The approximation $\bar{\theta}_N$ converges in probability to the integral θ as $N \to \infty$.*

Proof. The proof is a straightforward application of the preceding lemma. Let $\varepsilon, \eta > 0$ be given, and choose N_0 sufficiently large such that $\sigma/\sqrt{N_0\eta} < \varepsilon$. Then clearly

$$\Pr\left(|\bar{\theta}_N - \theta| \geq \varepsilon\right) \leq \Pr\left(|\bar{\theta}_N - \theta| \geq \sigma/\sqrt{N_0\eta}\right) \leq \Pr\left(|\bar{\theta}_N - \theta| \geq \sigma/\sqrt{N\eta}\right)$$

for any $N \geq N_0$. Now set $t = \sigma/\sqrt{N_0\eta}$. Then by the Lemma,

$$\Pr\left(\left|\bar{\theta}_N - \theta\right| \geq t\right) \leq \frac{\sigma^2}{N}\frac{N_0\eta}{\sigma^2} \leq \frac{\sigma^2}{N}\frac{N\eta}{\sigma^2} = \eta.$$

Combining this and the previous result, we have

$$\Pr\left(|\bar{\theta}_N - \theta| \geq \varepsilon\right) \leq \eta$$

for $N \geq N_0$. This is equivalent to saying that, for all $\varepsilon > 0$,

$$\lim_{N\to\infty} \Pr\left(|\bar{\theta}_N - \theta| \geq \varepsilon\right) = 0.$$

We have thus shown that $\bar{\theta}$ converges to θ in probability as $N \to \infty$. □

From the proof of Theorem 2.4 we can deduce that the error $|\bar{\theta}_N - \theta|$ is $O(1/\sqrt{N})$. In other words, the error decreases like $1/\sqrt{N}$ as $N \to \infty$. This should seem like a very slow rate of convergence. However, there is one particular advantage: this error estimate is independent of the dimension of the underlying space. This essentially means that we can obtain similar levels of accuracy with the same number of points, regardless of the dimension of the space.

We should note that the proofs of these two results are not necessarily critical to the content of this thesis. However, it is likely that the ideas used in these proofs can be applied in the future to the error analysis of the algorithm that will be presented here. It is for this reason that they have been included.

2.2.3 Implementation

While we have discussed the theoretical underpinnings of Monte Carlo integration, we have not given a concrete explanation of how one might implement the procedure on a computer. Such an algorithm could be easily deduced from the formulas that were derived in Section 2.2.1, but it would be worthwhile to give a step-by-step summary of a practical implementation. The following algorithm does this.

Algorithm 2.5. *Let $E \subset \mathbb{R}^n$ with finite Lebesgue measure, and let $f \in L(E)$. The integral of f over E can be estimated numerically using the following procedure:*

1. *Select an appropriate interval I which contains the set E.*

2. *Randomly select N points from a uniform distribution on I.*

3. *Evaluate the characteristic function χ_E at each of the random points to determine which ones lie in E.*

4. *Evaluate f at each point found in step 3.*

5. *Sum the function values found in step 4 and multiply by $|I|/N$.*

The value computed in the final step is an approximation to the integral.

We will conclude this section with a brief example which illustrates a straightforward application of Monte Carlo integration. This example may be familiar to some, as it computes an approximation to π using the procedure outlined in Algorithm 2.5.

Example 2.6. Consider the set

$$E = \left\{(x, y) \in \mathbb{R}^2 : x, y \geq 0 \text{ and } x^2 + y^2 \leq 1\right\}.$$

This is simply the portion of the unit disc which lies in the first quadrant. The true area of E is known to be $\frac{\pi}{4}$. We can approximate this area using Monte Carlo integration and use the result to estimate the value of π.

The area A of E can be found by integrating the constant function $f(x) = 1$ over the set E:

$$A = \int_E 1\,dx.$$

We will implement Algorithm 2.5 in MATLAB to estimate this integral numerically. An obvious choice for the interval $I \supset E$ is the unit square, $I = [0,1] \times [0,1]$. Given I, we randomly select $N = 10,000$ points from a uniform distribution on I. Figure 2.6 shows these points in relation to the sets E and I.

From these N points we select those which lie in the set E. That is, we evaluate the characteristic function of E, which is

$$\chi_E(x, y) = \begin{cases} 1 & x, y \geq 0 \text{ and } x^2 + y^2 \leq 1 \\ 0 & \text{otherwise} \end{cases}$$

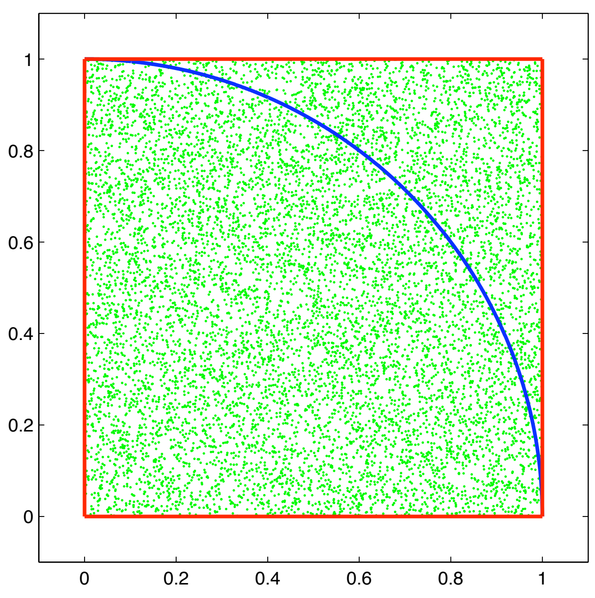


Figure 2.1. Region of integration for Example 2.6. The red lines denote the boundary of the interval I, and the set E is bounded between the x- and y-axes and the blue curve. The green dots represent the N points randomly sampled on I.

and disregard any points for which $\chi_E(x, y) = 0$. The results from MATLAB indicate that 7891 of the points lie in E. Since we are integrating a constant function, the integrand simply evaluates to 1 at each of these points. Therefore, per step 5 of Algorithm 2.5, an approximation to the area A is given by

$$\bar{A} = \frac{|I|}{N} \sum_{i=1}^{7891} 1 = \frac{7891}{10000} = 0.7891.$$

From this, we have

$$\pi \approx (4)(0.7891) = 3.1564.$$

While this is not a terrific approximation, it is reasonably close to the known value of π.

2.3 Monte Carlo Measure

In Example 2.6 we illustrated a method for obtaining an approximation to π. In doing so, we used Monte Carlo integration to estimate the area of a set in $\mathbb{R}^2$. However, the notion of "area" (or, more generally, "volume" or "hypervolume") is precisely Lebesgue measure. That is, given a Lebesgue-measurable set $E \subset \mathbb{R}^n$, we clearly have

$$|E| = \int_E 1\, dx.$$

As we alluded to in Section 2.1, this concept generalizes to arbitrary measures. Given a measure μ defined on $\mathbb{R}^n$, the μ-measure of E is

$$\mu(E) = \int_E 1\, d\mu. \tag{2.8}$$

This fact will allow us to utilize other measures for computing the onto defect.

Unfortunately, there is a slight problem. The machinery that we developed in Section 2.2 for implementing Monte Carlo integration depended heavily on the fact that we were integrating with respect to Lebesgue measure. The integral in (2.8) clearly does not fit this description, for it is an integral with respect to the measure μ. Thus it would seem that we are have pigeonholed ourselves into working exclusively with Lebesgue measure. This is not quite the case. We can employ other measures, provided that we place a certain restriction on them. This restriction will be based on the following definition, which is a slightly modified version of those found in Kolmogorov [8] and Wheeden and Zygmund [16].

Definition 2.7. *Let $(\mathcal{S}, \Sigma, \mu)$ be a measure space, and let ϕ be an additive set function defined on Σ. Then ϕ is said to be* ***absolutely continuous with respect to*** *μ if $\phi(E) = 0$ for every μ-measurable E for which $\mu(E) = 0$.*

Since any measure is itself an additive set function, we can apply this definition to the measures μ_1 and μ_2. The primary assumption that we will make regarding μ_1 and μ_2 is that they are absolutely continuous with respect to Lebesgue measure. Definition 2.7 itself is not particularly helpful in terms of understanding the importance of this assumption. However, the condition of absolute continuity allows us to invoke the following classical result from analysis, which will prove to be critical. The statement given here is based on those found in Rudin [11] and Wheeden and Zygmund [16].

Theorem 2.8 (Radon-Nikodym)**.** *Let $(\mathcal{S}, \Sigma, \mu)$ be a measure space, where μ is a σ-finite measure, and let ϕ be an additive set function defined on the σ-algebra Σ. If ϕ is absolutely continuous with respect to μ, then there exists a unique $f \in L(\mathcal{S}; d\mu)$ such that*

$$\phi(E) = \int_E f\, d\mu$$

for every measurable $E \subset \mathcal{S}$.

The beauty of this theorem is that it allows us to still interpret the measure of a set as a Lebesgue integral. That is, if a measure μ is absolutely continuous with respect to the Lebesgue measure on $\mathbb{R}^n$, then there exists a unique Lebesgue-integrable function ρ such that

$$\mu(E) = \int_E \rho(x)\, dx \tag{2.9}$$

for all μ-measurable $E \subset \mathbb{R}^n$. Thus we have converted (2.8) into an integral which is amenable to the Monte Carlo methods outlined in Section 2.2. Hence we can apply (2.7) to (2.9), which gives the approximation

$$\widetilde{\mu}(E) = \frac{|I|}{N} \sum_{i=1}^{N} \rho(X_i)\chi_E(X_i). \tag{2.10}$$

This equation expresses a Monte Carlo approximation to the μ-measure of E. It is still not quite ready to be implemented, however, as there are a few computational issues that must be addressed first.

2.4 Approximation of Characteristic Function

As mentioned previously, a major advantage of using Monte Carlo integration is that it only requires knowledge of the characteristic function of the given set. However, this can often prove to be problematic. In practice, very little is known about the sets D_2 and $f[D_1]$. This lack of knowledge often encompasses the characteristic functions as well. In this regard, it would seem that the techniques we have been developing will still prove to be difficult to implement computationally.

While we may not be able to write down an expression in closed form for these sets or their characteristic functions, we do have one computational weapon in our arsenal. We can, in general, generate random samples of points in the sets. Given such data, we can, in some sense, define an approximation to the characteristic function. This approximation can then be used to implement Monte Carlo integration.

Suppose we wish to measure the set $E \subset \mathbb{R}^n$, from which we generate the sequence of random points $E_0 = \{x_1, x_2, \ldots, x_N\}$. This random sample is a finite set, thus it has Lebesgue measure zero. Consequently, its characteristic function χ_{E_0} vanishes almost everywhere in I, so a Monte Carlo algorithm that replaces χ_E with χ_{E_0} in (2.10) would almost surely return a value of $\widetilde{\mu}(E) = 0$. This is fine if E is itself a set of measure zero, but one would hope that the algorithm would work for any measurable set E. Therefore, we cannot simply substitute χ_{E_0} for χ_E; we must use E_0 to construct an appropriate surrogate to χ_E. We will develop this approximate characteristic function by "fattening" the set E_0 by some specified amount and using the characteristic function of the resulting, larger set. This technique is inspired by a concept called natural measure, which is introduced in [2].

2.4.1 Natural Measure

Alligood, Sauer, and Yorke [2] introduce a technique for measuring sets which they term "rain gauge" measure. More specifically, they measure a set S using the rain gauge measure associated with a map f. Given an initial iterate x_0, the measure of S is defined as the fraction of the iterates of the orbit of x_0 under f which lie in S. Formally, this fraction is defined as

$$F(x_0, S) = \lim_{n\to\infty} \frac{\operatorname{card}\left(\left\{f^i(x_0) \in S : 1 \le i \le n\right\}\right)}{n}. \tag{2.11}$$

In order to avoid confusion with the notation that we have been using for Lebesgue measure, we have adopted the notation card$(\cdot)$ to denote the cardinality of a given set, as used by Halmos [4]. This concept should look somewhat similar to the previously discussed methods for measurement of a set by Monte Carlo integration. In fact, Lebesgue measure can be thought of as an example of rain gauge measure where the map f is replaced by a uniform random number generator, as the authors mention.

This informal notion of rain gauge measure is not sufficient, however, and problems arise quickly. In particular, the authors consider the case when S is a chaotic attractor associated with the map f. Suppose we take an initial iterate x_0 which lies in the basin of attraction of S, but not in S itself. Then $f^n(x_0)$ approaches S as $n \to \infty$, but $f^n(x_0) \notin S$ for any finite n. This would imply that

$$F(x_0, S) = \lim_{n\to\infty} \frac{0}{n} = 0.$$

In reality, however, the limit in (2.11) should approach 1, since there is some finite N for which $n \ge N$ implies that $f^n(x_0)$ is arbitrarily close to S. We can reconcile this issue by considering a neighborhood of S rather than S itself. That is, given $r > 0$, we denote the neighborhood of radius r around S as

$$N(r, S) = \{x : \operatorname{dist}(x, S) \le r\}$$

and study the fraction $F(x_0, N(r, S))$. The natural measure generated by the map f, or f-measure, is then defined to be

$$\mu_f(S) = \lim_{r\to 0} F(x_0, N(r, S)),$$

where we require that S be closed and almost all x_0 give the same result. Thus we have solved the problem of measuring S by enlarging it slightly and measuring the resulting set instead.

2.4.2 Fattening E_0

Now that we have discussed the concept of natural measure, we can return to the original problem of measuring E using the random sample E_0. While the issues that were mentioned regarding natural measure are quite different from those that we face, there are some fundamental similarities. These similarities will provide inspiration and motivation for techniques that will hopefully resolve our problem.

In constructing natural measure, we dealt with a case in which iterates of an orbit did not fall in a set, but instead came arbitrarily close to it. A similar phenomenon occurs if we attempt to deploy Monte Carlo integration on the set E_0. The probability of a randomly chosen point landing in E_0 is 0, but it is likely that such points will fall reasonably close to the points in E_0. Thus it would seem that we could remedy the situation by "fattening" E_0 like we did with the attractor S.

As the authors of [2] do with the attractor S, we will enlarge E_0 by a single, fixed amount. That is, we will place an open ball of radius r around each point in E_0. However, we will make one slight modification to the fattening procedure in this case. Rather than taking the limit as $r \to 0$, we will simply choose an appropriate radius r and fatten the set by that amount. A procedure for selecting this radius will be discussed in more detail in Chapter 3.

The resulting fattened set can be written as

$$E_r = \bigcup_{i=1}^{N} B_r(x_i) \tag{2.12}$$

The characteristic function of this set is given by

$$\chi_{E_r}(x) = \begin{cases} 0 & \text{if } ||x - x_m|| \geq r \\ 1 & \text{if } ||x - x_m|| < r, \end{cases} \tag{2.13}$$

where m is chosen such that

$$||x - x_m|| = \min \{||x - x_1||, ||x - x_2||, \ldots, ||x - x_N||\}.$$

The above expression simply states that x_m is the nearest neighbor of x in E_0. This definition may seem somewhat awkward at first, but it will allow for very easy and efficient evaluation of the function χ_{E_r} in MATLAB, as we will see later.

Given the fattened set E_r, we can use its characteristic function χ_{E_r} in place of χ_E in (2.10), which yields the formula

$$\widetilde{\mu}(E) = \frac{1}{N} \sum_{i=1}^{N} \rho(X_i) \chi_{E_r}(X_i). \tag{2.14}$$

This expression gives us a way of approximating the measure of a given set E using only a random sample from E. It is precisely this formula that we will use in practice when we attempt to compute measures.

2.5 Summary

Let us conclude this chapter by reviewing what we have accomplished so far and offering a preview of the following chapter. Thus far we have introduced the concept of Monte Carlo integration and discussed its applicability to our problem. We have also determined the assumptions that need to be made in order to make our problem suitable for Monte Carlo integration. In particular, we are now working under the assumption that the measure with which we would like to work is absolutely continuous with respect to Lebesgue measure. This work has culminated in a precise formula, shown in (2.14), which will allow us to approximate the measures of given subsets of $\mathbb{R}^n$. In Chapter 3 we will use this formula in practice as we deploy the algorithm on a collection of carefully chosen benchmark sets. This will hopefully give us insight into the problem of selecting the fattening radius r, while it will also allow us to gauge the efficacy of the algorithm.

Chapter 3

Analysis of Numerical Results

In this chapter, we will explore results obtained from deploying the algorithm of Chapter 2 on a variety of sets. We will attempt to determine the appropriate choice of the fattening radius r by studying the behavior of the calculated measure as r is varied. By analyzing these results for sets whose Lebesgue measure is known, we will hopefully be able to determine techniques with which to extract useful information from practical implementations of the algorithm.

3.1 Benchmark Examples

We will begin by describing several benchmark sets which we will use to gauge the utility of this algorithm. The numerical results that are presented later in the chapter will all come from implementations of the algorithm for these specific sets. For this discussion, we will group the sets into several different categories based on the complexity of their structures. The first eight sets will be primarily categorized based on their levels of connectedness.

3.1.1 Connected Sets

The first category will contain the simplest sets that we wish to study. Specifically, it will consist of connected subsets of $\mathbb{R}^n$. These will be either intervals or, for $n > 1$, less trivial connected subsets of $\mathbb{R}^n$.

The first two sets that we will consider are precisely intervals themselves. The first one is taken to be

$$E_1 = [0, 1/2],$$

which we will consider as a subset of the unit interval $[0, 1]$. By this we mean that we will choose the Monte Carlo integration interval to be $I = [0, 1]$. The second set is the two-dimensional analogue of E_1, defined as

$$E_2 = [0, 1/2] \times [0, 1/2].$$

As with E_1, we will consider E_2 as a subset of the unit interval in $\mathbb{R}^2$, $[0, 1] \times [0, 1]$. In both cases we have intentionally chosen the Monte Carlo interval I to be artificially large. This will allow us to study the behavior of the algorithm when the radius r grows unnecessarily large.

For our third example, we would like to consider a connected set which is not an interval. A simple example would be the unit disc in $\mathbb{R}^2$, so we will take

$$E_3 = \left\{(x, y) \in \mathbb{R}^2 : x^2 + y^2 \leq 1\right\}.$$

Unlike the previous two examples, we will choose the Monte Carlo interval to be the smallest interval containing E_3, which is $[-1, 1] \times [-1, 1]$.

The three examples just mentioned are depicted in Figures 3.1 and 3.2. Here we have

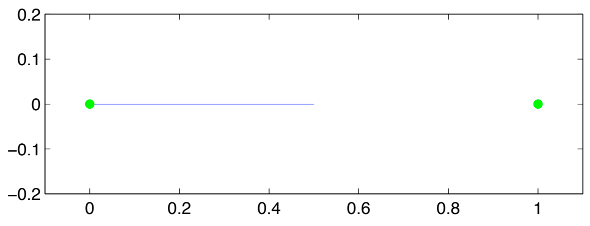
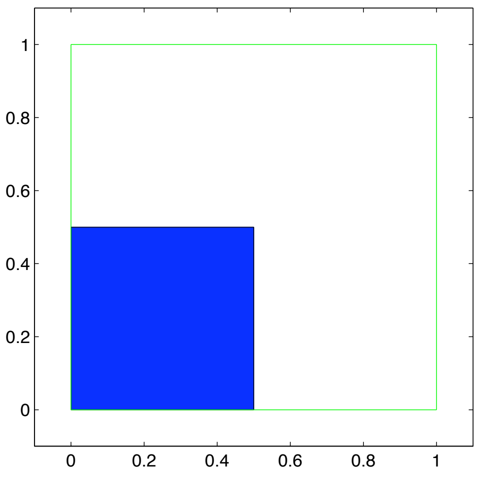
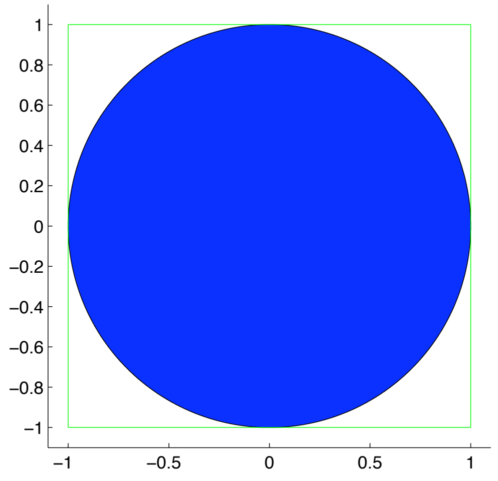


Figure 3.1. The benchmark set $E_1 = [0, 1/2]$. The filled blue line represents the set, and the green dots denote the endpoints (or boundary) of the Monte Carlo interval.

denoted the set of interest by with a filled blue region, and the boundary of the Monte Carlo interval is outlined in green. In all future examples, we will maintain this convention.



(a) $E_2 = [0, 1/2] \times [0, 1/2]$



(b) E_3, the unit disc in $\mathbb{R}^2$.

Figure 3.2. The sets E_2 and E_3. The filled region (blue) represents the set, and the outline (green) shows the boundary of the Monte Carlo interval.

3.1.2 Mildly Disconnected Sets

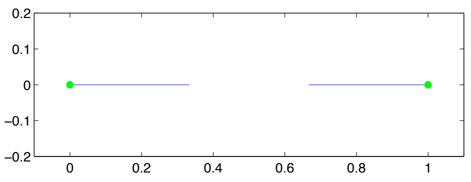
Rather than simply studying intervals or otherwise connected sets, we would also like to consider sets which exhibit some level of disconnectedness. Specifically, we will first look at sets that consist of a small number of nontrivial connected components. As a one-dimensional example we will consider

$$E_4 = [0, 1/3] \cup [2/3, 1].$$

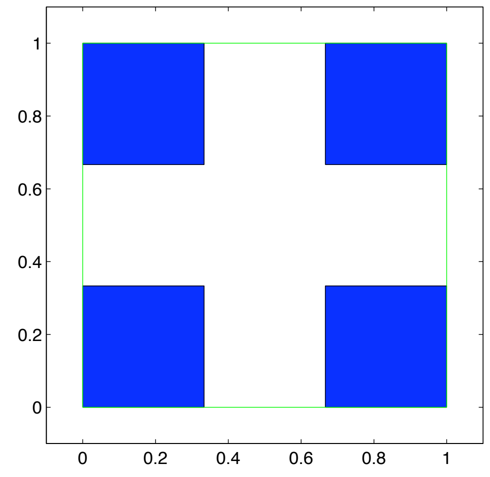
This set is a union of two disjoint intervals, hence it has two connected components. As with E_3, we will choose the Monte Carlo interval to be the smallest interval containing E_4, which is $[0, 1]$. Similarly, we will consider the two-dimensional analogue

$$E_5 = ([0, 1/3] \cup [2/3, 1]) \times ([0, 1/3] \cup [2/3, 1]),$$

embedded in the Monte Carlo interval $[0, 1] \times [0, 1]$. We see that this set is a union of four disjoint two-dimensional intervals, thus it has four components. This can be observed in Figure 3.3, which shows the sets E_4 and E_5.



(a) $E_4 = [0, 1/3] \cup [2/3, 1]$



(b) $E_5 = E_4 \times E_4$

Figure 3.3. The sets E_4 and E_5.

3.1.3 Highly Disconnected Sets

Extending the idea of the examples in the previous section, we would like to consider examples of sets which are highly disconnected. These are sets which contain a large number of components, and thus have a large number of "holes" at many different scales. Important examples of such sets are Cantor sets, which are actually totally disconnected[1]. We will be primarily studying sets of this form.

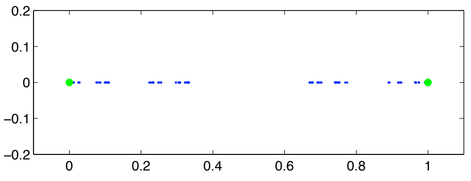
We will denote the next of our benchmark examples E_6, and let $E_6 = C$, which is the standard middle-third Cantor set. This set is quite well known, and detailed constructions of it can be found in Rudin [10] and Wheeden and Zygmund [16]. We will also consider the two-dimensional version,

$$E_7 = C \times C,$$

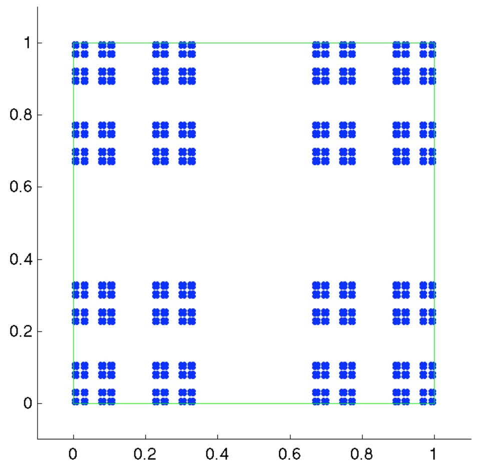
the so-called "Cantor dust." For both cases we will take the Monte Carlo interval to be the unit interval in the appropriate space. These two Cantor sets can be seen in Figure 3.4.

It is well-known that both E_6 and E_7 have Lebesgue measure zero, so these will provide useful tools for determining the accuracy of the algorithm when applied to measure-zero sets. However, it would also be helpful to test the algorithm on a set which has the same fundamental structure as the Cantor set, but with positive Lebesgue measure. Such sets

[1]A set is *totally disconnected* if its only connected subsets are one-point sets.



(a) $E_6 = C$



(b) $E_7 = C \times C$

Figure 3.4. E_6 and E_7, the middle-third Cantor set and the middle-third Cantor dust.

exist, and we will take the canonical example to be our next benchmark set. We will let E_8 be the standard "fat" Cantor set, sometimes referred to as the Smith-Volterra-Cantor set (and oftentimes abbreviated as SVC). This set is homeomorphic to the middle-third Cantor set and thus exhibits many of the strange topological properties of C. In particular, it is totally disconnected and nowhere dense in $[0, 1]$. However, unlike the Cantor set, the SVC has positive Lebesgue measure. Specifically, it has measure $1/2$. We omit the details of its construction, but a general description for a set of this type can be found in Royden [9] or Wheeden and Zygmund [16]. An illustration of the SVC can be seen in Figure 3.5.

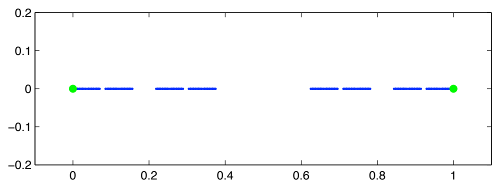
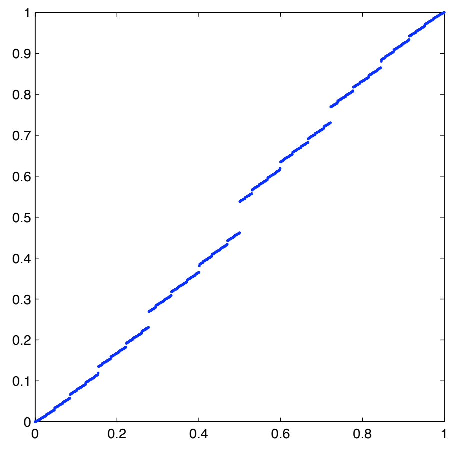


Figure 3.5. E_8, the Smith-Volterra-Cantor set.

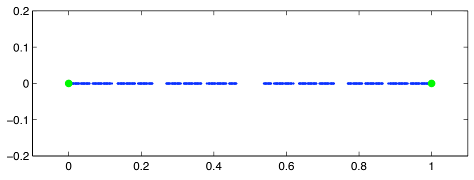
3.1.4 Commuters

The ultimate goal of this work is to measure the homeomorphic defect of a commuter function. With this in mind, it would be quite helpful to test the algorithm on a collection of such functions. In particular, we will attempt to measure the ranges of such functions. We will consider two particular classes of commuters: those between two symmetric tent maps of different heights, and those between a symmetric tent map and a skew tent map.

In the first case, we will take g_2 to be the standard tent map of height one, described in Section 1.2, and g_1 will be a symmetric tent map of height 0.9. This commuter, which we will denote by f_{E_9}, is shown in Figure 3.6 alongside its range, which is depicted as a subset of $[0, 1]$. The maps g_1 and g_2 are not conjugate in this case, so f_{E_9} should exhibit some



(a) f_{E_9}



(b) $E_9 = f_{E_9}([0, 1])$

Figure 3.6. The commuter f_{E_9}, evaluated at 1,000 points on $[0, 1]$ (left), and its range $f_{E_9}([0, 1]) \subseteq [0, 1]$ (right).

amount of homeomorphic defect. In particular, there are visible gaps in the range which signal that the function has a positive onto defect.

In the second case, we will again take g_2 to be the standard tent map. However, we will now make g_1 a skew tent map of height one whose peak occurs at the point $x = 1/4$. Figure 3.7 shows the resulting commuter, $f_{E_{10}}$, and its range as a subset of the unit interval. For future reference, let us mention that in this case, g_1 and g_2 are actually conjugate. As a result, $f_{E_{10}}$ is in fact a homeomorphism.

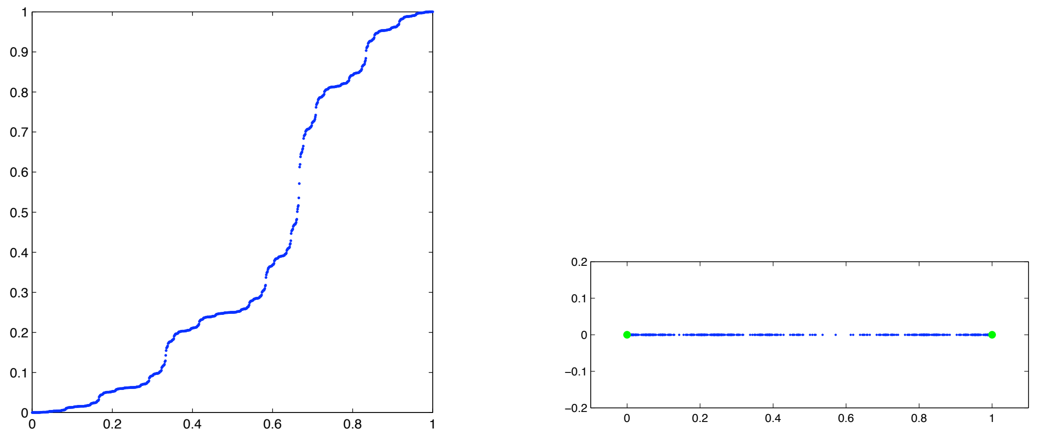


Figure 3.7. The commuter $f_{E_{10}}$, evaluated at 1,000 points on $[0,1]$ (left), and its range $f_{E_{10}}([0,1]) \subseteq [0,1]$ (right).

3.1.5 Summary of Examples

Before we begin presenting numerical results for these examples, we will briefly summarize by giving a list of the benchmark examples and certain important details. Table 3.1.5 gives the name of each example, a brief description, and the Lebesgue measure of each set (if it is known). This table is designed to give the reader a compact description of the collection of benchmark sets, along with some of the more critical properties of the sets.

3.2 Determination of Fattening Radius

We are now prepared to investigate the behavior of the Monte Carlo measure as the fattening radius is varied. This will hopefully allow us to determine criteria for selecting an appropriate radius for a given application. Throughout this section we will be considering graphs which depict the computed measure as a function of the fattening radius for each of the benchmark sets. We will restrict ourselves to the simple case of computing the Lebesgue measure of each set. Let us start by considering the first category of benchmark sets, the connected sets E_1, E_2, and E_3.

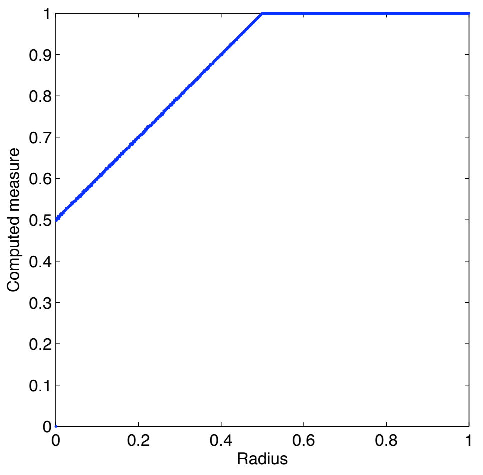
Set	Description	Measure
E_1	$[0, 1/2]$	$1/2$
E_2	$[0, 1/2] \times [0, 1/2]$	$1/4$
E_3	$\{(x, y) : x^2 + y^2 \leq 1\}$	π
E_4	$[0, 1/3] \cup [2/3, 1]$	$2/3$
E_5	$E_4 \times E_4$	$4/9$
E_6	Cantor set (C)	0
E_7	Cantor dust ($C \times C$)	0
E_8	Smith-Volterra-Cantor set (SVC)	$1/2$
E_9	$f_{E_9}([0, 1])$	unknown
E_{10}	$f_{E_{10}}([0, 1])$	1

Table 3.1. List of benchmark examples for numerical testing. Included is the name of the set, its definition, and its Lebesgue measure, if known.

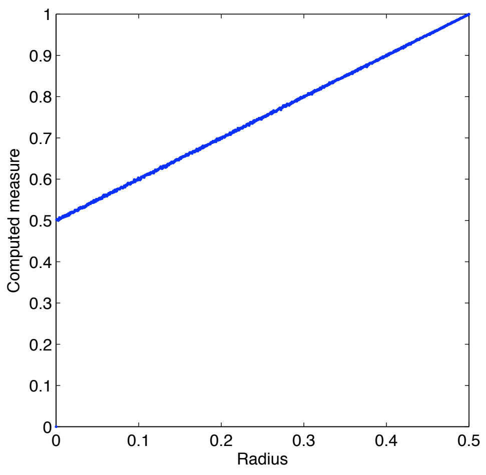
3.2.1 Results: Connected Sets

First, we will consider the set E_1. Figure 3.8 shows a series of graphs at different resolutions for this set. In Figure 3.8(a), we have allowed r to vary between 0 and 1. Observe that the graph undergoes two sharp transitions. The first occurs when the computed measure is approximately $1/2$, which we know to be the true measure of the set. We will refer to this event as the *first saturation*, as it is the point at which the fattened set has covered, or saturated, E_1. The second transition appears when the measure reaches 1, which is the full measure of the Monte Carlo interval $[0, 1]$. This is the point at which the Monte Carlo interval becomes saturated, so as with the first case we will refer to it as the *second saturation*. The computed measure remains constant after the second saturation. This behavior is to be expected, as the fattened set has already covered the entire Monte Carlo interval, so all randomly selected points will fall within the fattened region.

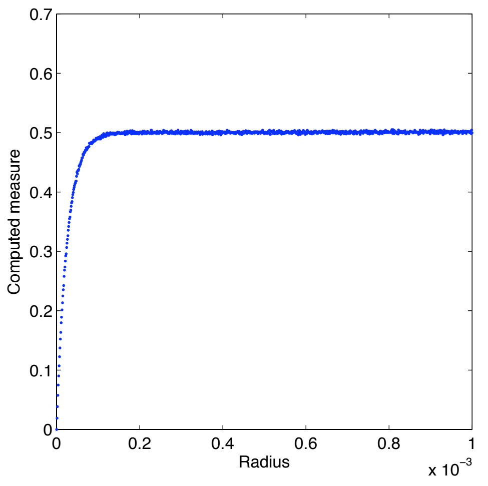
We would like to investigate the behavior of the computed measure prior to the second saturation, which occurs when $r \approx 1/2$. Figure 3.8(b) shows this, with $0 \leq r \leq 1/2$. Observe that the measure appears to grow linearly as a function of r between the first and second saturations. However, it is still not clear how the measure behaves prior to the first saturation, which occurs when r is extremely small. In Figure 3.8(c) we see a closer view of this region. We have still not completely honed in on the first saturation, but we have a slightly clearer view of it. It appears that the transition is not exactly sharp at this small



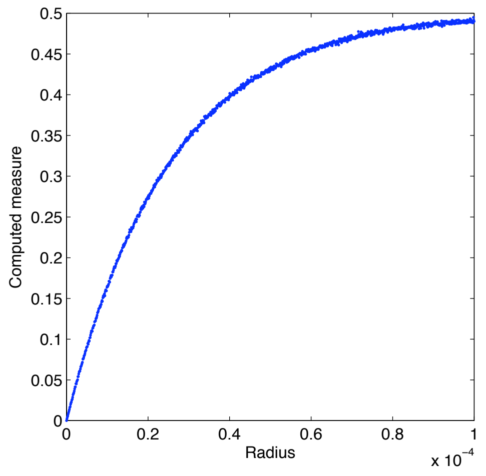
(a) $0 \leq r \leq 1$



(b) $0 \leq r \leq 0.5$



(c) $0 \leq r \leq 0.001$

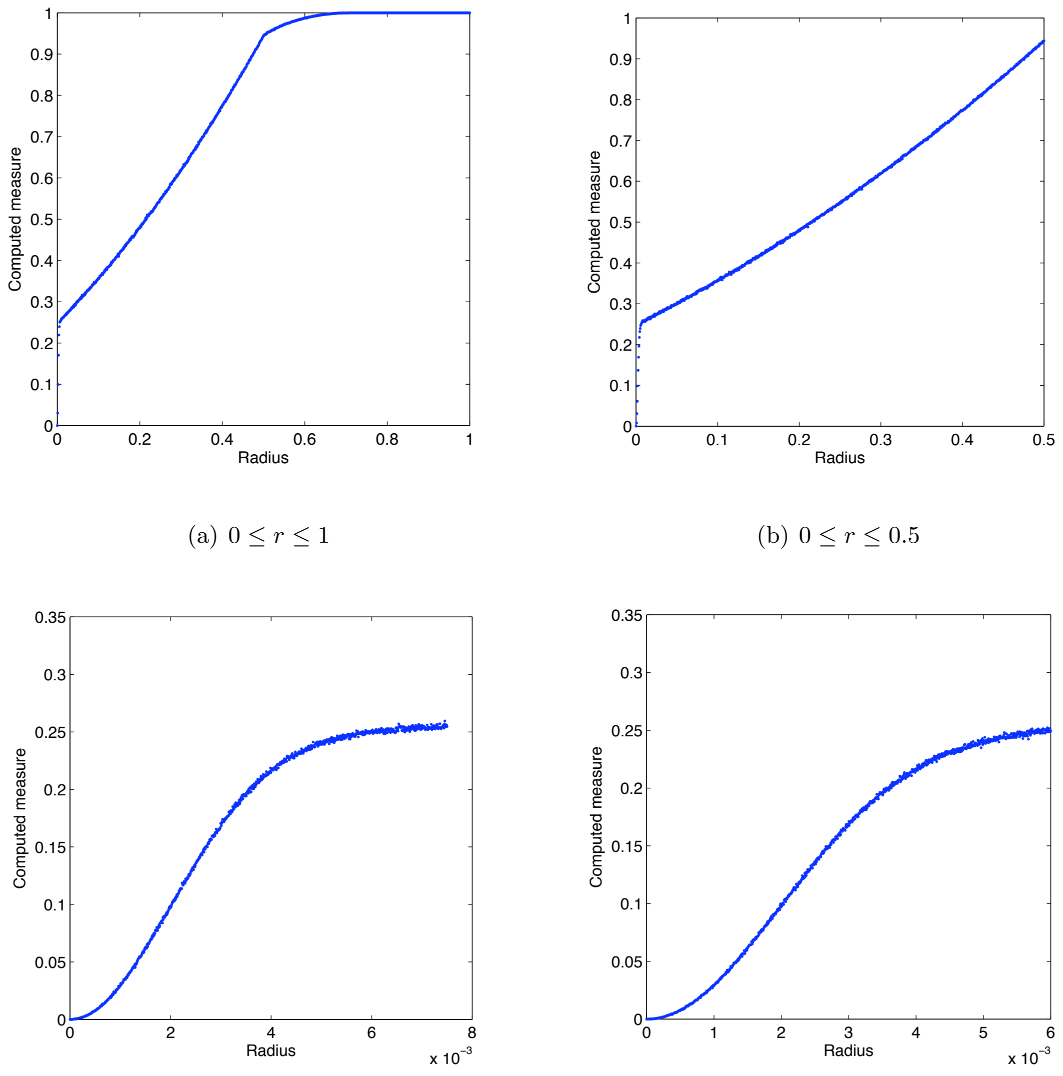


(d) $0 \leq r \leq 0.0001$

Figure 3.8. Graphs of computed Monte Carlo measure versus fattening radius for the set E_1. Each was created using a sample of 10,000 points randomly selected from E_1, 100,000 Monte Carlo points, and 1,000 values of r, evenly spaced on the specified interval.

scale, but is somewhat more gradual. However, given that this transition corresponds to a change in r of approximately 10^{-4}, the change is quite abrupt at a macroscopic scale. A closer view of the transition can be seen in Figure 3.8(d).

We will now consider a similar set of graphs for the two-dimensional analogue E_2. These are shown in Figure 3.9. In Figure 3.9(a) we see that, once again, two sharp transitions



(a) $0 \leq r \leq 1$

(b) $0 \leq r \leq 0.5$

(c) $0 \leq r \leq 0.0075$

(d) $0 \leq r \leq 0.006$

Figure 3.9. Graphs of computed Monte Carlo measure versus fattening radius for the set E_2. Each was created using a sample of 10,000 points randomly selected from E_1, 100,000 Monte Carlo points, and 1,000 values of r, evenly spaced on the specified interval.

are exhibited. The first occurs when the computed measure is approximately 1/4, which is the true measure of E_2. Thus this transition corresponds to the first saturation. However, the second transition does not quite coincide with the second saturation. Note that the transition occurs when $\mu \approx 0.95$, while the full measure of the Monte Carlo interval is 1. Additionally, after the transition μ approaches 1 gradually, which is again unlike the behavior seen with E_1. Needless to say, this behavior is quite strange, but it can be explained relatively easily. We will address this issue in Section 3.3.

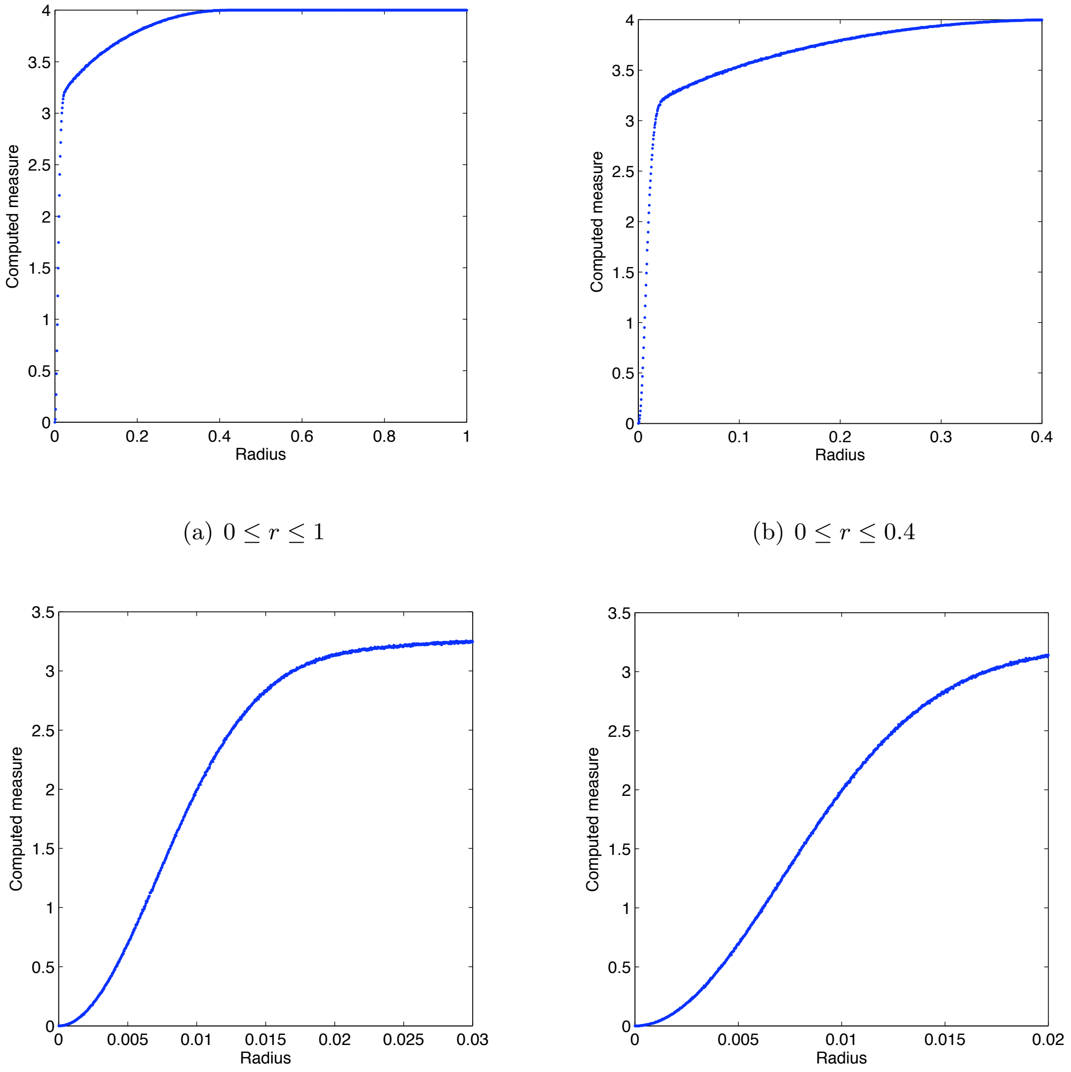
If we consider the graph prior to the second transition, the behavior looks much more similar to that which we have already seen with E_1. Figure 3.9(b) shows this region, with $0 \leq r \leq 1/2$. Observe that μ grows smoothly with r after the first saturation, albeit not in a linear fashion. This likely represents a correlation between the growth and the dimension of the underlying space. Also, Figures 3.9(c)-(d) show a closer view of the first saturation. Once again, we see that the sharp transition is actually a smooth one occurring over a very small range of r values.

Finally, a collection of graphs for E_3 is shown in Figure 3.10. The behavior seen here should be somewhat similar to that seen for E_1 and E_2. In particular, there is a sharp transition corresponding to the first saturation at $\mu \approx \pi$, which is the known measure of E_3. However, there is no second transition, as μ simply grows gradually to 4 after the first saturation. This is actually remarkably similar to the behavior seen after the second transition in Figure 3.9(a) for E_2. A closer look at the growth in this regime is seen in Figure 3.10(b). Additionally, Figures 3.10(c)-(d) show a better view of the behavior of μ in the vicinity of the first saturation. As with the previous two cases, the sharp transition is actually a smooth change taking place over a very small change in r.

3.2.2 Results: Mildly Disconnected Sets

We will now turn to the mildly disconnected sets E_4 and E_5. First, Figure 3.11 shows a series of measure-radius graphs for E_4. Notice that all four plots look remarkably similar, at least qualitatively, to those shown for the previous one-dimensional set E_1. In Figure 3.11(a), we see that there are two sharp transitions, the first at $\mu \approx 2/3$ and the second at $\mu \approx 1$. These correspond to the first and second saturations, respectively. Figure 3.11(b) demonstrates the linear growth of μ between the first and second saturations, as was observed with E_1. Figures 3.11(c)-(d) show closer views of the first saturation, which again manifests itself as a smooth transition at a very small scale.

Just as E_4 demonstrated similar behavior to E_1, the behavior of E_5 bears a striking resemblance to that of the first two-dimensional example, E_2. Figure 3.12 shows this.



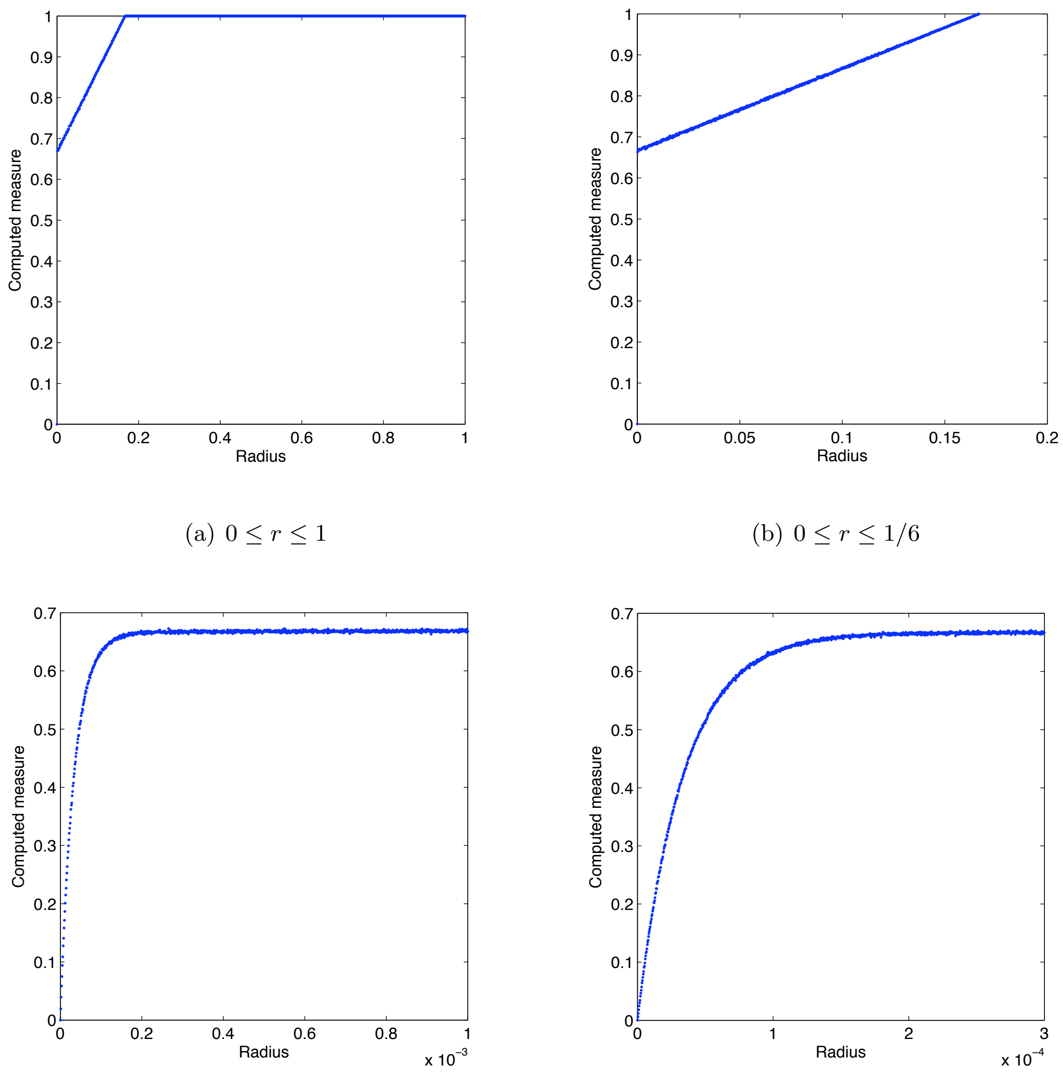
(a) $0 \leq r \leq 1$

(b) $0 \leq r \leq 0.4$

(c) $0 \leq r \leq 0.03$

(d) $0 \leq r \leq 0.02$

Figure 3.10. Graphs of computed Monte Carlo measure versus fattening radius for the set E_3. Each was created using a sample of 10,000 points randomly selected from E_1, 100,000 Monte Carlo points, and 1,000 values of r, evenly spaced on the specified interval.



(a) $0 \le r \le 1$

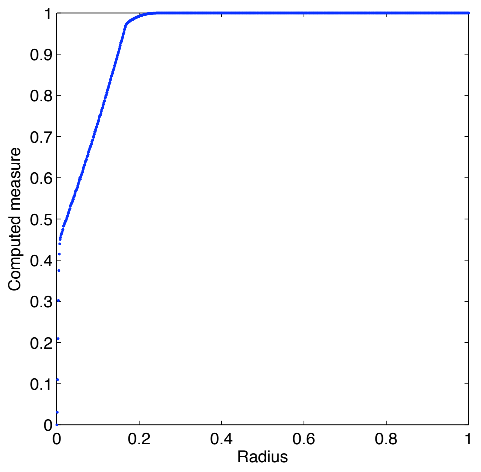
(b) $0 \le r \le 1/6$

(c) $0 \le r \le 0.001$

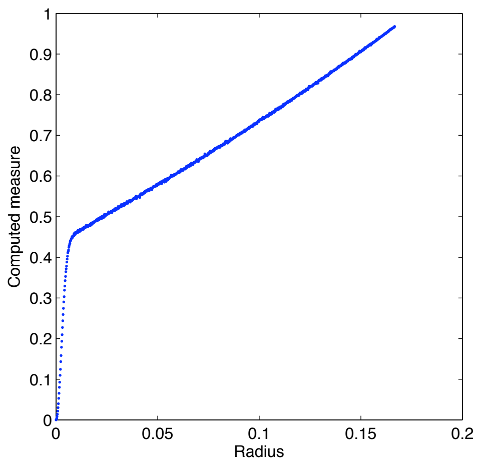
(d) $0 \le r \le 0.0003$

Figure 3.11. Graphs of computed Monte Carlo measure versus fattening radius for the set E_4. Each was created using a sample of 10,000 points randomly selected from E_4, 100,000 Monte Carlo points, and 1,000 values of r, evenly spaced on the specified interval.

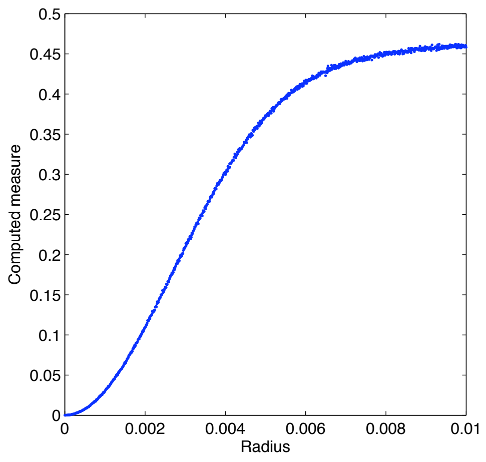
Figure 3.12(a) looks quite similar to Figure 3.9(a), again showing two sharp transitions.



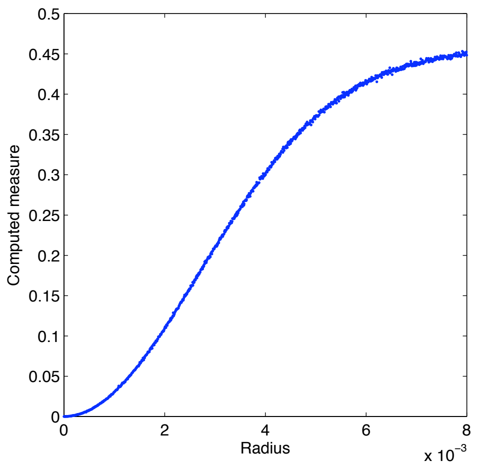
(a) $0 \le r \le 1$



(b) $0 \le r \le 1/6$



(c) $0 \le r \le 0.01$



(d) $0 \le r \le 0.008$

Figure 3.12. Graphs of computed Monte Carlo measure versus fattening radius for the set E_5. Each was created using a sample of 10,000 points randomly selected from E_5, 100,000 Monte Carlo points, and 1,000 values of r, evenly spaced on the specified interval.

The first occurs near $\mu = 4/9$ and coincides with the first saturation. The second occurs just prior to the second saturation, with μ growing gradually toward 1 after the transition. Figure 3.12(b) shows the smooth, yet nonlinear, growth of μ between the first and second transitions. A close view of the first saturation is shown in Figures 3.12(c)-(d), which look nearly identical to Figures 3.9(c)-(d), albeit on a different scale.

Thus far, it appears that there is a recurring theme with each of the sets we have tested.

While the behavior of μ as a function of r has certainly varied greatly, in each case we have seen a sharp transition corresponding to the first saturation. That is, the growth of the computed measure μ as a function of r changes fundamentally when μ is near the true measure of the set. This observation is very promising, and it would be quite advantageous if this behavior is replicated in the remaining examples.

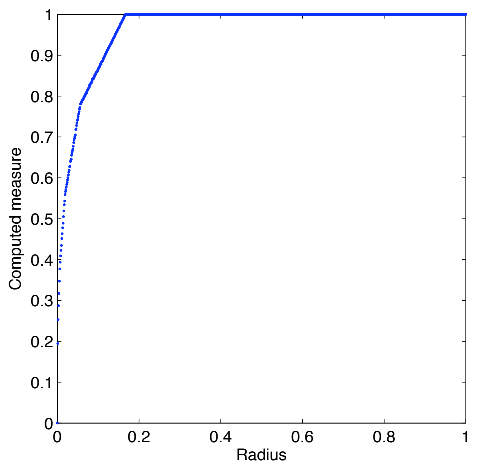
3.2.3 Results: Highly Disconnected Sets

We are now ready to test the algorithm on a class of much more complex sets. These sets will be represented by a collection of highly disconnected sets, namely the totally disconnected Cantor sets E_6, E_7, and E_8.

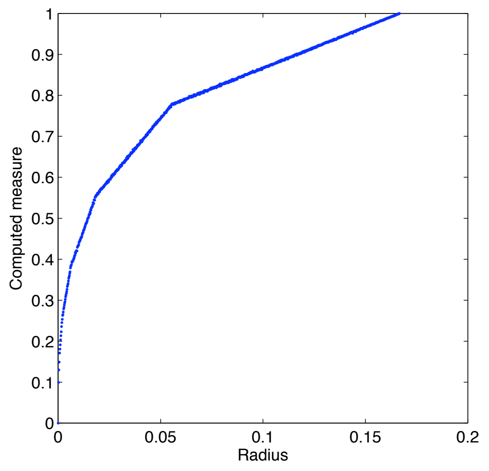
Figure 3.13 shows the measure-radius plots for E_6, the standard middle-third Cantor set. It should be apparent that these graphs demonstrate behavior that is quite different from anything seen in the previous examples. The one visible similarity is the linear growth of μ between transitions, which is reminiscent of the one-dimensional examples E_1 and E_4. However, there are several sharp transitions that appear, as opposed to the two that were seen previously. Moreover, these transitions appear at all scales, as we can see in Figures 3.13(b)-(d). In fact, it appears that they occur at $r = 1/6$, $r = 1/18$, $r = 1/54$, and, in general, at $r = 1/(2 \cdot 3^n)$ for $n \in \mathbb{Z}^+$. We will discuss the reason for this phenomenon in Section 3.3.

In Figure 3.14, we see that E_7, the Cantor dust, exhibits behavior similar to that of E_6. Once again, there are sharp transitions at all scales, and they appear to be occurring at the same values of r as in the previous case. The lone similarities to the earlier cases are the smooth nonlinear growth of μ between transitions and the behavior of μ after the final transition. Both are similar to the respective behaviors of E_2 and E_5.

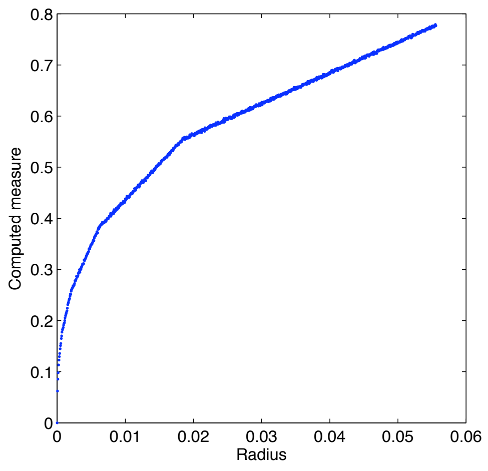
Now we turn to E_8, the Smith-Volterra-Cantor set. A series of measure-radius graphs for this set are shown in Figure 3.15. Observe that Figures 3.15(a)-(b) look similar to the corresponding figures for the middle-third Cantor set. There are multiple sharp transitions, with linear growth between them. Figures 3.15(c)-(d) still exhibit multiple transitions, but the behavior for very small values of r looks different from that seen in Figures 3.13(c)-(d). It appears that μ grows very rapidly with r when r is extremely small. Figure 3.15(e) confirms this, showing a sharp transition when $\mu \approx 1/2$. Finally, Figure 3.15(f) shows the transition to be smoother at a small scale, as we have seen in several of the previous cases. Overall, it appears that the algorithm has managed to detect the fact that $|E_8| = 1/2$, despite the fact



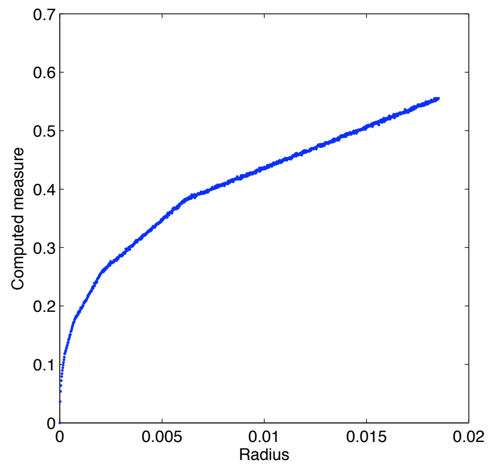
(a) $0 \leq r \leq 1$



(b) $0 \leq r \leq 1/6$

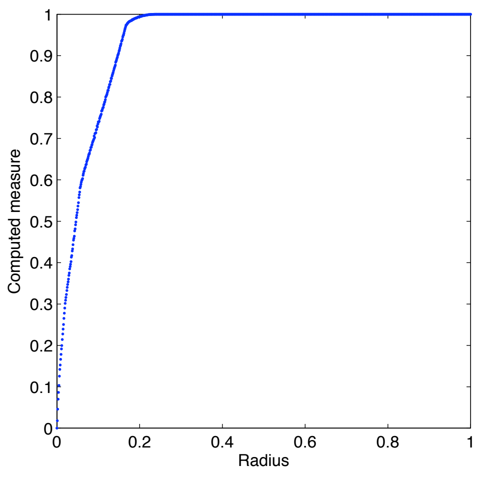


(c) $0 \leq r \leq 1/18$

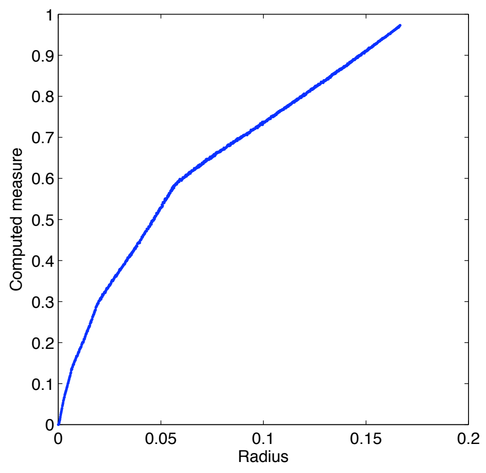


(d) $0 \leq r \leq 1/54$

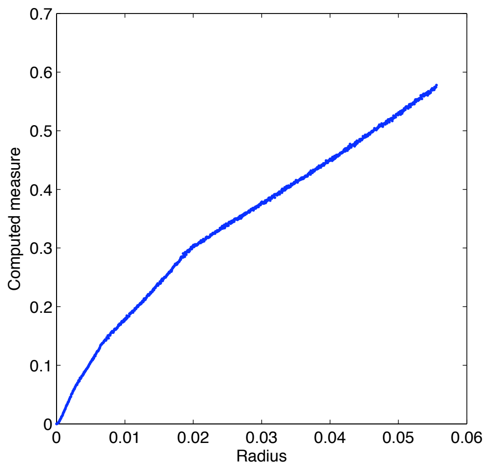
Figure 3.13. Graphs of computed Monte Carlo measure versus fattening radius for the set E_6. Each was created using a sample of 10,000 points randomly selected from E_6, 100,000 Monte Carlo points, and 1,000 values of r, evenly spaced on the specified interval.



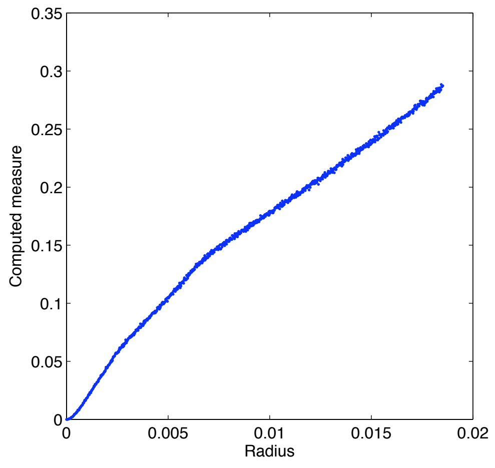
(a) $0 \leq r \leq 1$



(b) $0 \leq r \leq 1/6$

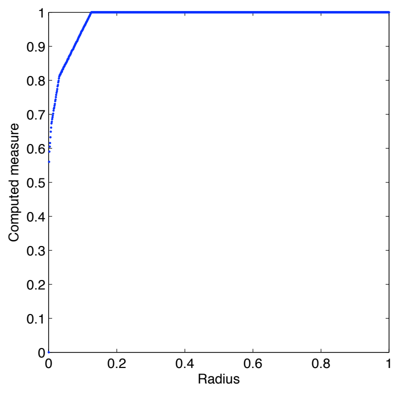


(c) $0 \leq r \leq 1/18$

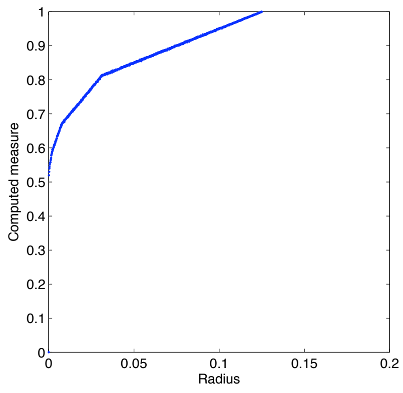


(d) $0 \leq r \leq 1/54$

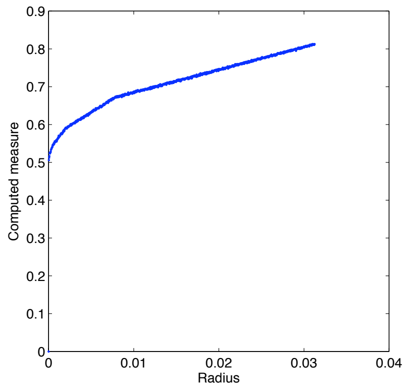
Figure 3.14. Graphs of computed Monte Carlo measure versus fattening radius for the set E_7. Each was created using a sample of 10,000 points randomly selected from E_7, 100,000 Monte Carlo points, and 1,000 values of r, evenly spaced on the specified interval.



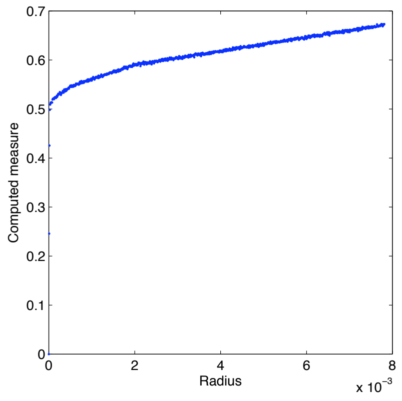
(a) $0 \leq r \leq 1$



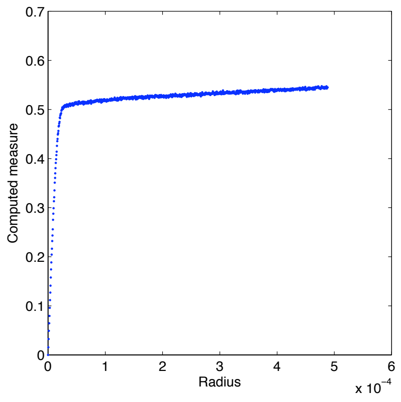
(b) $0 \leq r \leq 1/8$



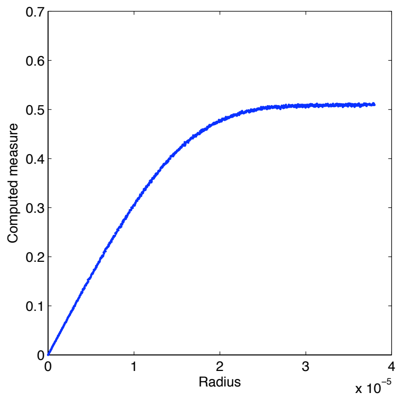
(c) $0 \leq r \leq 1/32$



(d) $0 \leq r \leq 1/128$



(e) $0 \leq r \leq 1/2048$



(f) $0 \leq r \leq 3.8 \times 10^{-5}$

Figure 3.15. Computed measure versus radius for E_8. Each was created using 2^{14} sample points, 100,000 Monte Carlo points, and 1,000 values of r.

that E_8 has the same fundamental structure as the Cantor set. This observation provides a very promising argument for the efficacy of the algorithm.

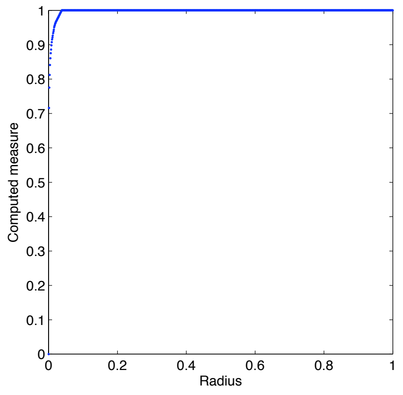
3.2.4 Results: Commuters

We will now discuss numerical results for the final class of benchmark sets, the commuters. First, we will consider E_9, the range of a commuter between two symmetric tent maps. A series of graphs for this set is shown in Figure 3.16. Observe that there is a sharp transition corresponding to the second saturation. However, the growth of μ prior to that is somewhat reminiscent of the middle-third Cantor set, albeit more smooth. This behavior is exhibited in each of six figures, and it appears that there is no sharp transition corresponding to the first saturation. If the algorithm behaves as it did in the previous cases, we could conclude that E_9 has Lebesgue measure zero. However, it is not entirely apparent whether we can form such a conclusion with any level of certainty.

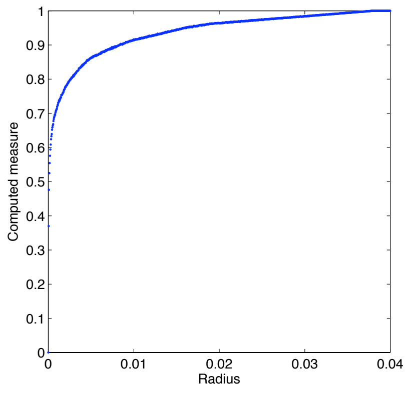
Graphs of measure versus radius for E_{10} are shown in Figure 3.17. Note that the behavior seen in each graph is very similar to that shown in the analogous graph for E_9. There is a sharp transition corresponding to the second saturation, but the growth is largely smooth prior to that. This is somewhat problematic, since the commuter that we are studying is actually a conjugacy. As a result, it maps $[0, 1]$ onto $[0, 1]$, so the Lebesgue measure of E_{10} should be 1.

Recall that in most previous cases the Monte Carlo interval strictly contained the set we were attempting to measure. With this in mind, we can attempt to resolve the problem by making the Monte Carlo interval slightly larger than $[0, 1]$. We will choose the interval $[-1/2, 3/2]$. A new set of graphs for this interval is shown in Figure 3.18. Here, as before, we see a sharp transition that coincides with the second saturation. However, we also see some strange behavior near $\mu = 1$. The behavior of μ changes abruptly here, albeit not as suddenly as in previous cases. Also, it appears that the transition occurs just prior to $\mu = 1$. This can likely be explained by the fact that the sample of E_{10} has noticeable gaps in it. If we refer back to Figure 3.7, we see that there are some places where the graph of the function is nearly vertical, so evaluation of the function on a finite sample will inevitably produce some gaps in the range.

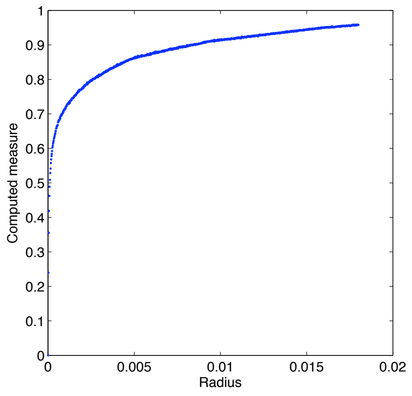
This example is not nearly as convincing as the previous ones, but it still appears somewhat promising. If we set aside the unknown cases of commuters, it seems that the algorithm is reasonably effective overall at approximating the Lebesgue measure of a set



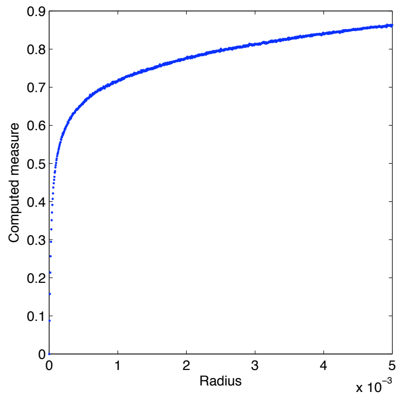
(a) $0 \leq r \leq 1$



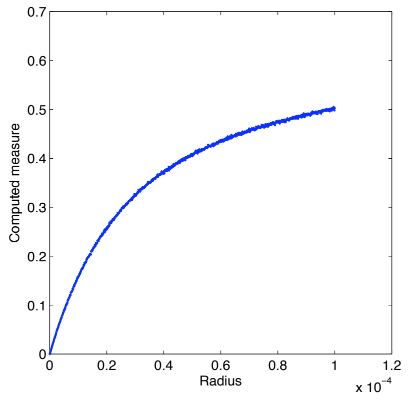
(b) $0 \leq r \leq 0.04$



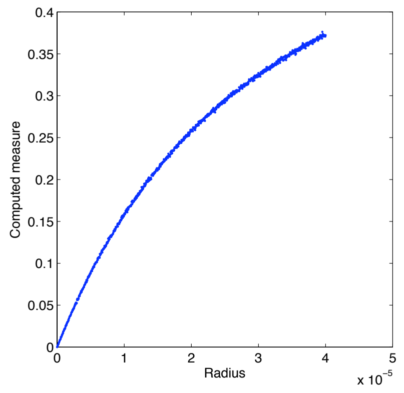
(c) $0 \leq r \leq 0.018$



(d) $0 \leq r \leq 0.005$

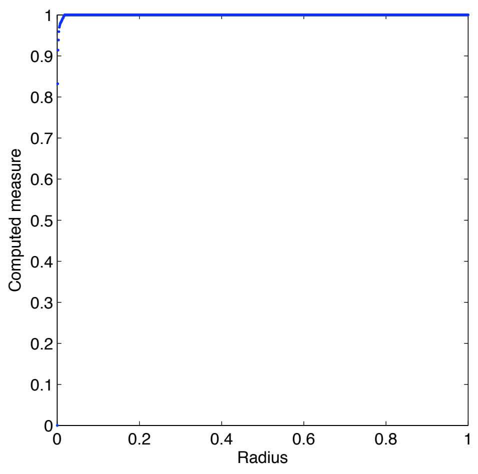


(e) $0 \leq r \leq 0.0001$

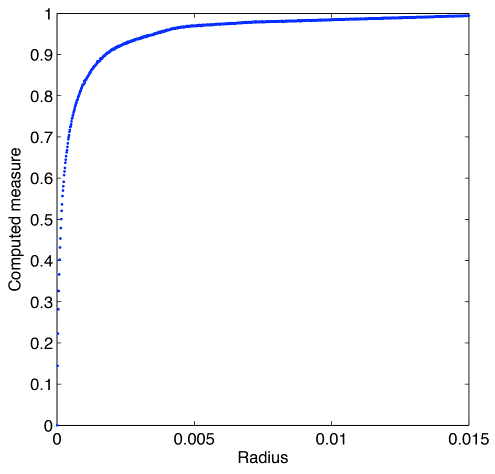


(f) $0 \leq r \leq 0.00004$

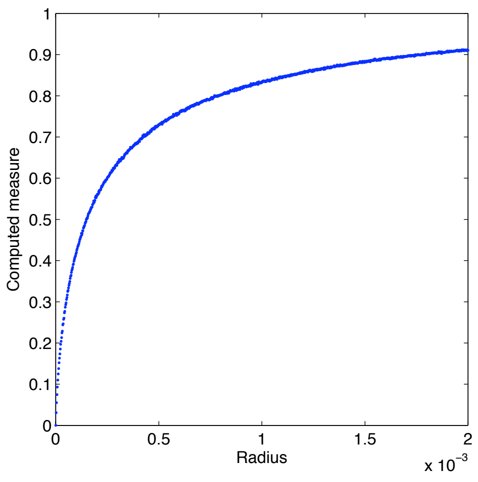
Figure 3.16. Computed measure versus radius for E_9. Each was created using 10000 sample points, 100,000 Monte Carlo points, and 1,000 values of r.



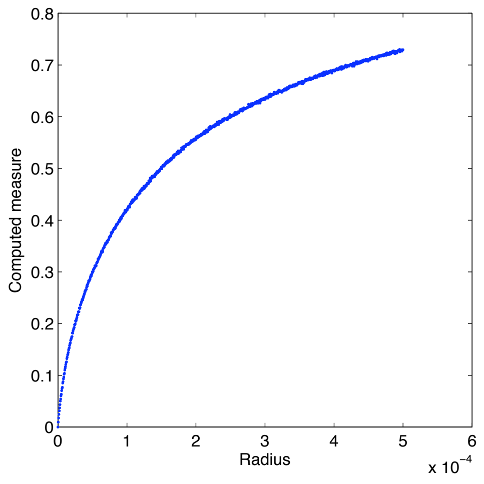
(a) $0 \leq r \leq 1$



(b) $0 \leq r \leq 0.015$

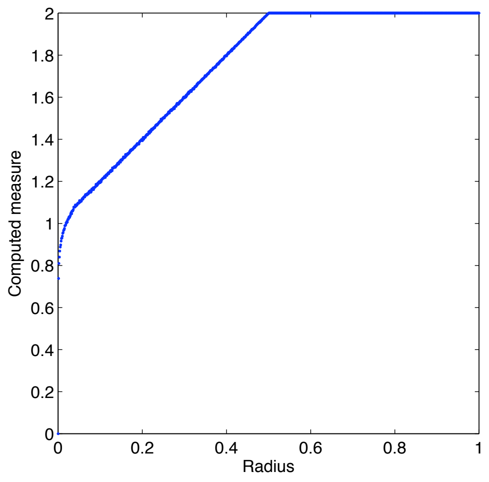


(c) $0 \leq r \leq 0.002$

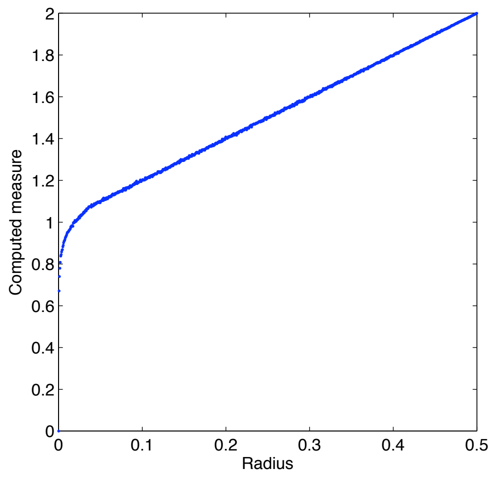


(d) $0 \leq r \leq 0.0005$

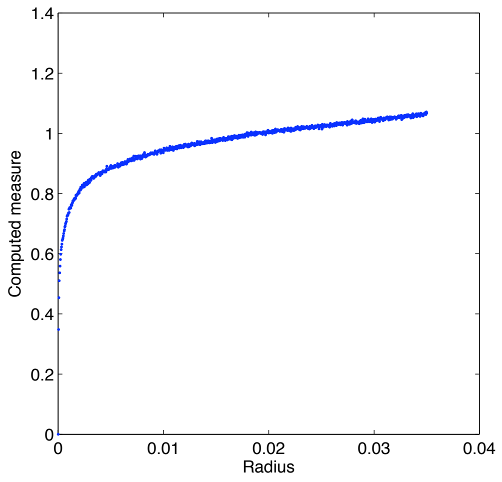
Figure 3.17. Computed measure versus radius for E_{10}. Each was created using 10000 sample points, 100,000 Monte Carlo points, and 1,000 values of r.



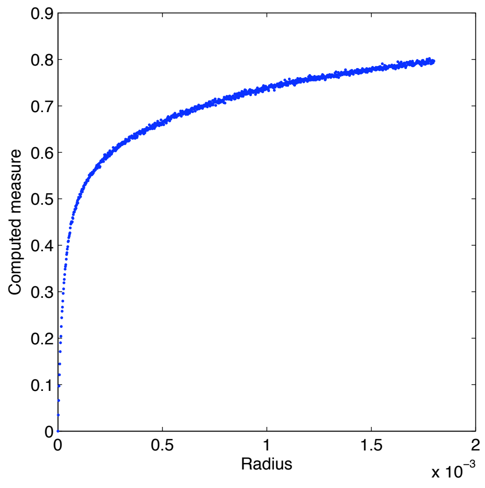
(a) $0 \leq r \leq 1$



(b) $0 \leq r \leq 0.015$



(c) $0 \leq r \leq 0.002$



(d) $0 \leq r \leq 0.0005$

Figure 3.18. Computed measure versus radius for E_{10}. Each was created using 10000 sample points, 100,000 Monte Carlo points, and 1,000 values of r.

based on a finite sample of that set. We will expand on this discussion in Chapter 4, as well as discuss future work for improving the algorithm.

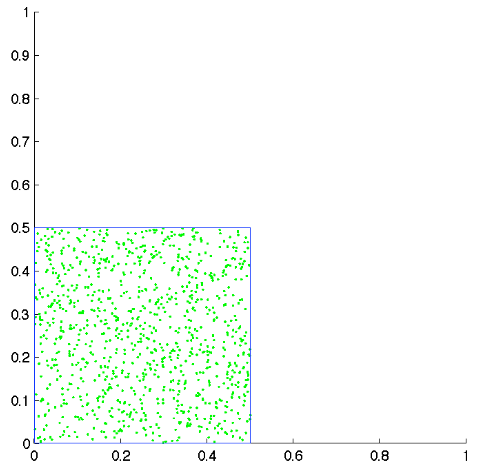
3.3 Discussion of Behavior

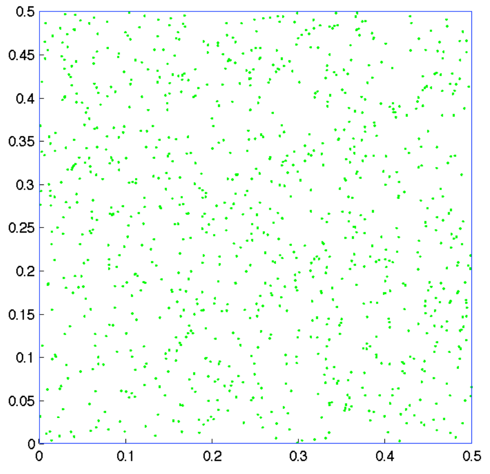
Now that we have studied plots of computed measure versus fattening radius for various benchmark sets, we will attempt to explain the behavior that we have seen. We will by no means give a rigorous explanation for this behavior, rather we will give a reasonable heuristic argument.

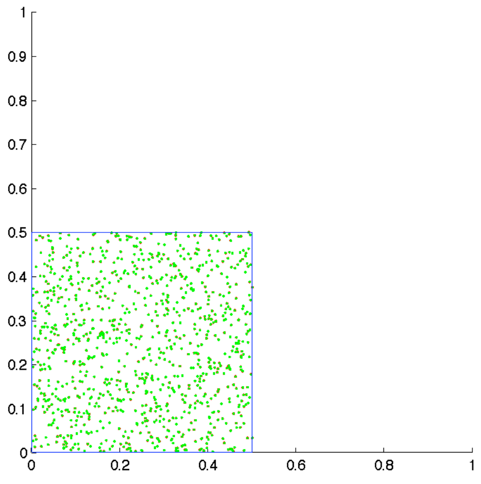
Among the examples that we have studied, there have been some fundamental similarities in the growth of μ as a function of r. In all cases, μ grows somewhat rapidly at first, and then the growth changes fundamentally. In some cases, this change is very abrupt, while in others it is quite gradual. After the change, μ grows more slowly as it approaches the full measure of the Monte Carlo interval. We will discuss the reasons that underlie these specific behaviors. Throughout this section, we will refer to Figures 3.19 and 3.20, which show a step-by-step illustration of the fattening process for the set E_1. In each of these figures we have used 1,000 sample points in order to make the images more readable, as opposed to the 10,000 points that were used for the actual numerical simulations.

When r is extremely small, as in Figures 3.19(a)-(b), the open balls do not overlap. This results in very rapid growth of μ as r increases. As the balls begin to overlap, the growth slows. This stage in the fattening process is shown in Figures 3.19(c)-(d) and 3.20(a). Once the balls have completely overlapped, the growth no longer slows, and μ appears to be proportional to r^n, where n is the dimension of the underlying space. This should be apparent from Figures 3.20(b) and (c), where we see that the fattened set has now taken on the approximate shape of square with a rounded corner, along with mildly distorted edges. Given this illustration, it seems reasonable that μ should grow like r^2 in this regime. Finally, Figure 3.20(d) shows the fattened set for a large value of r, in which case almost all of the Monte Carlo interval has been covered.

Let us now focus on the range of r values between the points where the balls initially overlap and where they have completely overlapped. In the case of E_1, this range is very small. In fact, this transition from no overlap to complete overlap corresponds to the sharp transition that is seen near $\mu = 1/2$. This likely results from the fact that the sample points generated on E_1 are taken from a uniform distribution. Because of this, the variation in


(a) $r = 10^{-6}$


(b) Zoom of $r = 10^{-6}$


(c) $r = 10^{-5}$

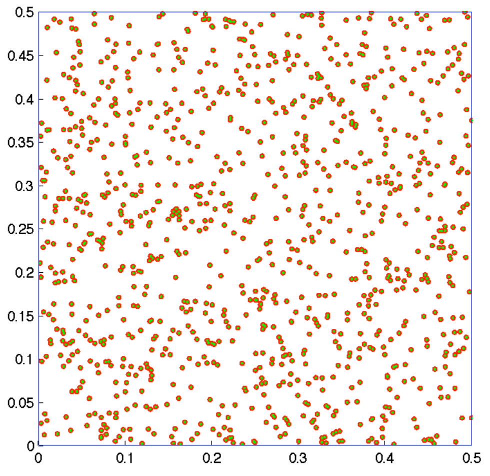
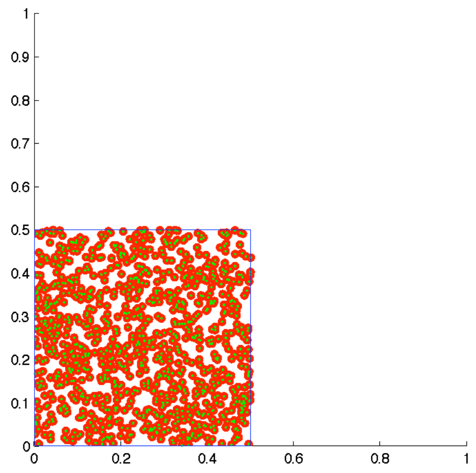
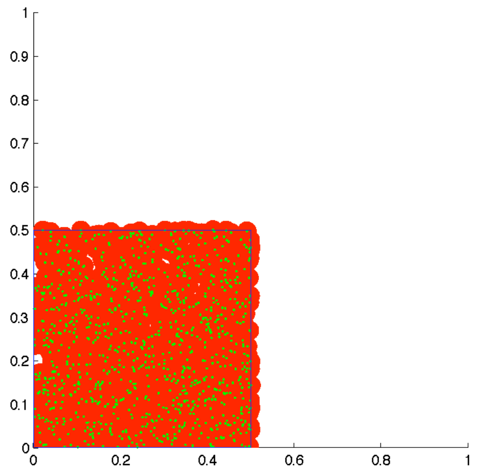

(d) Zoom of $r = 10^{-5}$

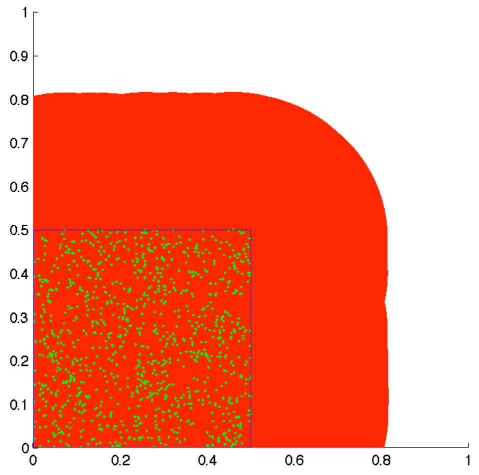
Figure 3.19. Depiction of fattened set $\widetilde{E}$ for different values of r. Figure (a) shows the set for $r = 10^{-6}$, where the balls are all still disjoint. Figure (b) shows a closer view for $r = 10^{-6}$. Figure (c) shows the set for $r = 10^{-5}$, where the balls have just begun to overlap. Figure (d) shows a closer view for $r = 10^{-5}$.



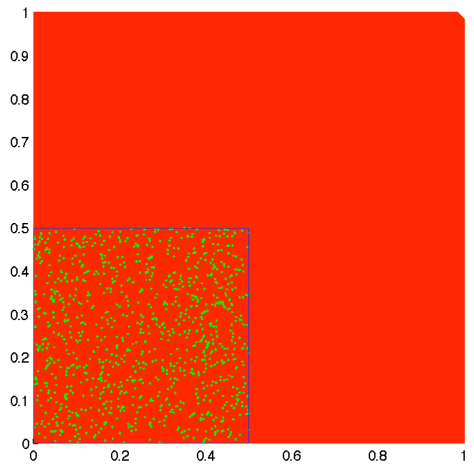
(a) $r = 10^{-4}$



(b) $r = 5 \times 10^{-4}$



(c) $r = 0.1$



(d) $r = 0.5$

Figure 3.20. Depiction of fattened set $\widetilde{E}$ for different values of r (continued). Figure (a) shows the set for $r = 10^{-4}$, where the balls have now begun to overlap considerably. In Figure (b), $r = 5 \times 10^{-4}$, and the set being measured has been essentially covered. Note that there is also some overflow, where the balls have expanded slightly beyond the boundary of the set. In Figure (c), $r = 0.1$, and the balls have expanded well beyond the set. Note that the covered set somewhat resembles a square with one rounded corner containing the interval. In Figure (d), $r = 0.5$, and the balls have grown large enough to essentially cover the entire unit interval.

the sizes of the gaps between points should be relatively small, so the gaps are all covered quickly and at approximately the same time. This results in the sharp transition that we previously witnessed.

We can also use these arguments to explain the strange behavior that was seen near the second saturation for E_2. Recall that Figure 3.20(c) shows the fattened set to be a square with a rounded corner after the first saturation. The set maintains this shape as it continues to grow. However, a point is eventually reached where the sides of this square overlap the edges of the unit interval, while the corner of the interval still has not been completely covered. It is at this point that the growth fundamentally changes, as evidenced by the sharp transition just prior to the second saturation in the case of E_2.

This explanation that we have just detailed can also be applied to sets for which the transition is much more gradual, such as E_6, the Cantor set. The middle-third Cantor set has gaps at all scales, so the transition from no overlap to complete overlap is much more gradual. In fact, there will be a transition in the growth of μ near $r = 1/(2 \cdot 3^n)$ for each integer n. This is precisely half the width of the intervals that are removed in the n^{th} step of the construction of the Cantor set. Near this value of r, the balls centered at points bordering the gap will merge, covering the gap and slowing the growth of μ. This phenomenon also explains the large number of transitions that appeared in the measure-radius graph for E_6, and the fact that they seemed to arise at all scales.

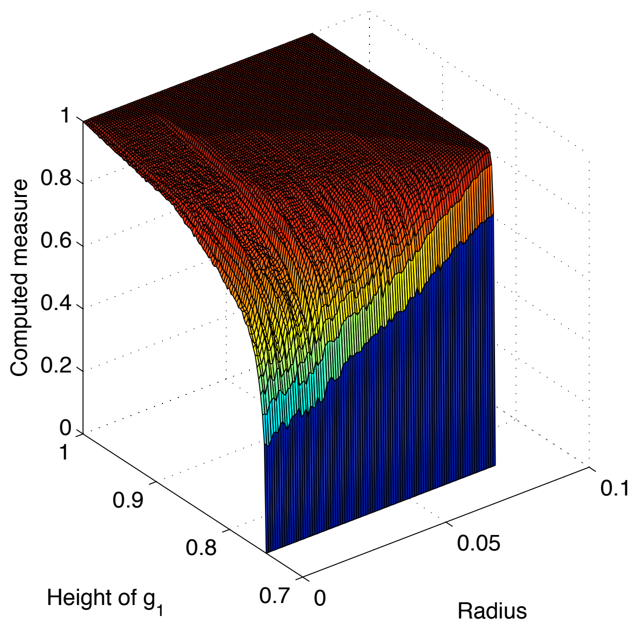
As previously mentioned, this section does not give a rigorous explanation of the behavior seen in Section 3.2. It is simply a discussion, undertaken with the intention of improving our understanding of the algorithm. This will hopefully aid in the further development of the algorithm, as well as its effective use in the future.

3.4 Sequences of Commuters

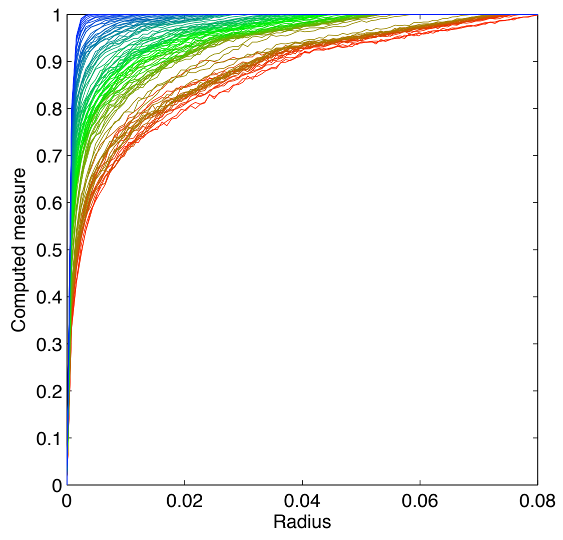
We will conclude this section on numerical experimentation with one final consideration regarding commuters. Given the unknown circumstances surrounding the deployment of the algorithm on commuters, it would be nice to be able to find a way to put the algorithm to good use. That is, even if it does not accurately approximate the true measures of these sets, we may be able to use it effectively as a tool for detecting homeomorphic defect.

A necessary condition for the algorithm to be a suitable surrogate for measuring onto defect is that it must be monotone with respect to the true onto defect. More specifically, when the onto defect increases, the surrogate should increase, and when the onto defect decreases, the surrogate should also decrease. A way to analyze such behavior would be to study sequences of commuters. In particular, we will study commuters between a symmetric tent map of height a_i and the standard tent map for some sequence $\{a_1, a_2, \ldots, a_n\}$ which increases monotonically toward 1. It is known that the measure of the range of the commuters should tend to 1 as this sequence approaches 1, so we would like to see if the computed measure does the same.

Figure 3.21 shows graphs related to such an experiment. Figure 3.21(a) shows a surface plot of the computed measure μ versus the fattening radius and the height of the map g_1. Observe that the computed measure increases with the fattening radius, as we should expect from our previous results. Also, it appears that the measure generally increases as the height of g_1 increases. Figure 3.21(b) gives a better look at this. In this figure, we have plotted the computed measure versus the radius for each height value. The colors of the corresponding lines change gradually from red to green to blue as the height changes. It appears that the computed measure does in fact increase with the height. However, there are some visible problems, as the behavior is not perfectly monotone. This could simply be due to the inherent randomness of the algorithm, but further investigation would be required to confirm this. We will discuss this result and its potential importance more in Chapter 4.



(a) Surface plot of computed measure versus radius and height.



(b) Curves showing measure versus radius for various height values.

Figure 3.21. Plots of computed measure versus fattening radius and height corresponding to a sequence of commuters. Here the commuters relate g_1 and g_2, where g_1 is a symmetric tent map of height strictly less than 1, and g_2 is the symmetric tent map of height 1. In (a) we have a surface plot of the computed measure as a function of heigh and radius. In (b), we have multiple plots of measure versus radius. Each curve corresponds to a particular height value, and the color changes from red to green to blue as the height increases.

Chapter 4

Conclusions

We will now conclude this thesis by discussing what we have learned and outlining possible future work on this topic.

4.1 Discussion of Results

Throughout this thesis we have worked to develop an algorithm for accurately determining the Lebesgue measure of an arbitrary measurable subset of $\mathbb{R}^n$. This work ultimately led to an algorithm that is based on the technique of Monte Carlo integration, which we discussed in detail in Chapter 2. We then tested this algorithm on a collection of benchmark sets with the hope of gaining insight into the behavior of the algorithm. The results were presented in Chapter 3, along with a discussion of the behavior that was observed. We are now prepared to elaborate on this discussion and summarize the knowledge that we have gained through these experiments.

Perhaps the most evident phenomenon that we were able to witness was the presence of a sharp transition corresponding to the first saturation in most of the sets that were considered. This is important because it signals that an approximation to the true measure of the set can be determined by simply locating the transition on the graph of computed measure versus fattening radius. We should stress the fact that this phenomenon was not visible for all the sets that were considered; there were a select few sets for which no sharp transition was witnessed in this regime. Included in these sets were the Cantor set and Cantor dust. These two examples should not be particularly troubling, as they are both known to have Lebesgue measure zero. Because of this, the first saturation occurs as soon as r becomes larger than zero, so the absence of a sharp transition is not surprising. Additionally, the algorithm appears to be effective at detecting the true measure of the Smith-Volterra-Cantor set, which has the same topological structure as the middle-third Cantor set. These observations together present a reasonable amount of evidence in favor of the validity of this algorithm.

The other sets for which sharp transitions were not seen are not so easily explained. We are specifically referring to the sets E_9 and E_{10}, which were derived from commuter functions. In the case of E_9, there was no sign of a sharp transition, which could either indicate that the set has measure zero, or the algorithm is ineffective for sets of this type. For E_{10}, a transition of some form was witnessed near the true measure, but the observation was not quite as convincing as with earlier examples. Given the lack of knowledge surrounding the properties of these functions, it is hard to judge whether these observations are meaningful, or if the algorithm needs to be altered to properly handle this class of sets. This is an issue that should be addressed in the future, and we will discuss it more in Section 4.2.

One final issue regarding commuters is that of sequences of commuters, as discussed in Section 3.4. In that section, we proposed that the algorithm could still be used as a suitable surrogate in the event that it is ineffective at accurately computing measures associated with commuters. This conjecture could be tested by studying a sequence of commuters and determining whether the output of the algorithm behaved in a monotone fashion. Based on the numerical results, it appears that the computed measure generally increases as the sequence progresses. However, there were clearly some issues with regard to monotonicity. It is likely that these problems have something to do with the probabilistic nature of the algorithm, but it is by no means certain that this is the cause. This is clearly another issue which should be investigated further in the future in order to determine the true nature of the underlying behavior.

4.2 Future Work

Despite seeing many promising results from this algorithm, there are still several apparent shortcomings and many unknowns surrounding its behavior. Because of this, it is clear that there is still much work that can be done on this topic in the future. In this section, we will list several issues that should be addressed, as well as certain tasks that should be undertaken in future work.

- **Perform error analysis.** In particular, work should be done to determine justifiable error estimates for the algorithm. The error arising from Monte Carlo integration could be easily determined by using Tchebyshev's inequality and the related results discussed in Section 2.2.2. However, the error associated with the construction of the fattened set $\widetilde{E}$ is likely to be very nontrivial, and will subsequently require a considerable amount of work to determine. Hopefully such work would lead to proofs of certain results regarding the algorithm, such as those related to convergence. Rigorous error analysis should also help to shed some light on the validity of the algorithm when it is applied to commuters.

- **Further investigate sequences of commuters.** In the previous section we discussed the issues surrounding sequences of commuters that were encountered in Section 3.4. In the future, work should be done to further investigate these issues, and attempt to explain the monotonicity problems that were seen. This could potentially be handled by performing additional carefully chosen numerical experiments, or the theoretical work done regarding error analysis could provide insight.

- **Test the algorithm with other absolutely continuous measures.** In all of the examples that we have considered, we have restricted ourselves to simply calculating the Lebesgue measure of the given set. However, the algorithm is designed to work with any measure which is absolutely continuous with respect to Lebesgue measure. It would likely be advantageous to test the algorithm on some of these measures and gauge the resulting behavior.

The following is a list of things that could be done in the future to improve the present algorithm and achieve some of the eventual goals of this work.

- **Write code to automatically compute measure.** By this we mean that the algorithm should be modified to automatically detect the sharp transition corresponding to the first saturation. Obviously this should only be done when the algorithm is well understood and its output is justifiably accurate.

- **Extend the algorithm to accomodate arbitrary measures.** The original intent of this work was to develop a procedure for measuring subsets of $\mathbb{R}^n$ using arbitrary measures. We have instead constructed an algorithm which requires that the measures be absolutely continuous with respect to Lebesgue measure. An important goal to work toward would be to remove this restriction and develop an improved algorithm that can handle arbitrary measures. It is not clear whether this can be done by building on the existing algorithm or if new techniques must be introduced.

- **Deploy the algorithm on the other defect measures.** The ultimate goal of this work is to be able to reliably compute the homeomorphic defect of a given commuter function. This would require the ability to measure each of the four defects that were mentioned in Chapter 1. We mentioned the onto defect as motivation for the work we have done, as it can be easily calculated once we have the ability to computationally measure sets. The other defect measures are less simple, and more work would need to be done in order utilize our algorithm on them. However, such an accomplishment is the goal of this research, and it should be pursued in the future.

Overall, this algorithm presents much promise for future research. There is a great deal of work that can and should be done to improve it, for which we have outlined several particular issues to be addressed. However, a strong foundation has been laid for future work on this topic which will hopefully lead to the ultimate goal of accurately computing homeomorphic defect.

Bibliography

[1] Myron B. Allen III and Eli L. Isaacson. *Numerical Analysis for Applied Science.* John Wiley & Sons, Inc., 1998.

[2] Kathleen T. Alligood, Tim D. Sauer, and James A. Yorke. *Chaos: An Introduction to Dynamical Systems.* Springer, 1996.

[3] Morris H. DeGroot and Mark J. Schervish. *Probability and Statistics.* Addison Wesley, third edition, 2002.

[4] Paul R. Halmos. *Naive Set Theory.* Springer, 1974.

[5] J.M. Hammersley and D.C. Handscomb. *Monte Carlo Methods.* John Wiley & Sons, Inc., 1965.

[6] Myron Hlynka and Deborah Loach. Generating uniform random points in a regular n-sided polygon. Technical Report WMSR 05-02, University of Windsor, Windsor, Ontario, Canada N9B 3P4, 2005.

[7] John L. Kelley. *General Topology.* Springer-Verlag, 1975.

[8] A.N. Kolmogorov and S.V. Fomin. *Introductory Real Analysis.* Dover Publications, 1975.

[9] Halsey Royden. *Real Analysis.* Prentice Hall, third edition, 1988.

[10] Walter Rudin. *Principles of Mathematical Analysis.* McGraw Hill, 1976.

[11] Walter Rudin. *Real and Complex Analysis.* McGraw Hill, 1986.

[12] Joseph D. Skufca and Erik M. Bollt. Relaxing conjugacy to fit modeling in dynamical systems. *Physical Review E*, 76(026220), 2007.

[13] Joseph D. Skufca and Erik M. Bollt. A concept of homeomorphic defect for defining mostly conjugate dynamical systems. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 18(013118), 2008.

[14] William R. Wade. *An Introduction to Analysis*. Prentice Hall, third edition, 2003.

[15] Stefan Weinzierl. Introduction to monte carlo methods. arXiv:hep-ph/0006269v1, 2000.

[16] Richard L. Wheeden and Antoni Zygmund. *Measure and Integral: An Introduction to Real Analysis*. CRC Press, 1977.

Appendix A

Code

This appendix contains the MATLAB code that has been used throughout the production of this thesis. The complete code for each program is given, along with a brief description of the program's function. We have divided the programs into several groups based on their purpose.

Monte Carlo Integration

The code in this section is used in the implementation of the Monte Carlo integration algorithm.

mc_int.m

This program uses Monte Carlo integration to approximate the measure of a set $E \subset \mathbb{R}^n$. It takes in a random sample of E, the bounds for the Monte Carlo integration interval, the number of points to be used for the integration, the density function for the measure to be used, and the fattening radius. The sample is fattened by the amount specified by r, and the fattened set is measured via Monte Carlo integration.

```
function measure=mc_int(sample, bounds, N, density, r)

% ****************
% mc_int.m
% Author: Scott M. LaLonde
% Date created: 2/12/08
% Last modified: 4/10/09
% ****************
%
% MC_INT        Computes the measure of a subset E of R^n using
%               Monte Carlo integration. This is done using a ran-
%               dom sample of E and a measure which is absolutely
%               continuous with respect to Lebesgue measure.
%
% Usage:        measure = mc_int(sample, bounds, N, density)
%
% Input:
%
%   sample      Random sample of size N from the set E. This
%               should be given as an N-by-n array; that is, the
%               rows of 'sample' correspond to points in R^n.
%
%   bounds      Upper and lower bounds of an n-dimensional inter-
%               val containing E. The lower bounds are contained
%               in the first column and the corresponding upper
%               bounds are contained in the second column. This
%               should be an n-by-2 array.
%
%   N           Number of random points to be used for Monte Carlo
%               integration. This should be a positive integer.
%
%   density     Handle for function relating the desired measure
%               to the Lebesgue measure, per the Radon-Nikodym
%               theorem.
%
%   r           Radius by which set is to be fattened. This should
%               be a non-negative scalar.
%
% Output:
%
%   measure     Approximate measure of the set E. This returns a
%               non-negative scalar.
%
% MC_INT computes an approximation to the measure of a subset E of
% R^n, where the measure is absolutely continuous with respect to
% Lebesgue measure. It takes in a random sample of points in E and
% a radius by which to fatten the sample. Monte Carlo points are
% taken from the n-dimensional interval defined by 'bounds'. The
% characteristic function of the fattened set is evaluated at
% these points by calling 'sample_ind.m', and these values are
% then plugged into the Monte Carlo integration formula to obtain
% the approximation 'measure'.
% ****************
% Select Monte Carlo points in the specified interval.
x=rand(N, size(bounds, 1));
x=x*diag(bounds(:,2)-bounds(:,1))+ones(N, 1)*bounds(:,1)';

% Evaluate characteristic function at MC points.
I=sample_ind(x, sample, r);

% Compute measure.
measure=sum(density(x(I > 0)));
measure=prod(bounds(:,2)-bounds(:,1))/N*measure;
```

sample_ind.m

This program constructs the characteristic function for a fattened set, given a sample and a fattening radius. The characteristic function is then evaluated at a specified number of points.

```
function I=sample_ind(x, sample, r)

% *****************
% sample_ind.m
% Author: Scott M. LaLonde
% Created 2/12/09
% Modified 4/10/09
% *****************
%
% SAMPLE_IND    Computes the indicator function of a finite set
%               that has been fattened by a specified radius.
%               Evaluates the function at a given set of points.
%
% Usage:        I=sample_ind(x, sample, r)
%
% Input:
%
%   x           Set of N points at which the function is to be
%               evaluated. This should be input as N-by-n array;
%               that is, rows of 'x' correspond to points in R^n.
%
%   sample      Sample of points of size M. This should be input
%               as an M-by-n array.
%
%   r           Radius by which 'sample' is to be fattened. Should
%               be input as a non-negative scalar.
%
```

```
% Output:
%
%   I           Values of indicator function at points in 'x'.
%               Each value will be either 1 or 0, depending on
%               whether the corresponding point is in the fattened
%               set or not. Output as an N-by-1 vector.
%
% SAMPLE_IND computes the values of the characteristic function of
% a fattened set at a specified set of points. For each point in
% 'x', the nearest neighbor to that point in 'sample' is deter-
% mined. These distances are then checked against the radius 'r';
% if the distance is less than 'r', the function evaluates to 1,
% and otherwise the value is zero.
% *****************

% Compute nearest neighbors in 'sample' for points of 'x', along
% with corresponding distances.
[neighbors, dists]=annquery(sample', x', 1);
dists=dists';

% Compare distances to radius 'r' to determine function value.
I=zeros(size(x));
I(dists <= r)=ones(length(find(dists <= r)), 1);
```

lebesgue.m

This code defines the density function for the Lebesgue measure, per the Radon-Nikodym theorem. It may appear trivial, as it is simply the constant function 1. However, by constructing the code in this way we have allowed for the easy deployment of other absolutely continuous measures.

```
function m=lebesgue(x)

% *****************
% legesgue.m
% Author: Scott M. LaLonde
% Date created: 9/23/08
% Last modified: 4/10/09
% *****************
%
% LEBESGUE      Evaluate the density function for Lebesgue measure
%               at a given set of points.
%
% Usage:        m=lebesgue(x)
%
```

```
% Input:
%
%   x           Vector of N points in R^n at which the function is
%               to be evaluated. This should be an N-by-n array;
%               that is, rows of 'x' correspond to points in R^n.
%
% Output
%
%   m           Values of the function at the points in 'x'. This
%               will be an N-by-1 array.
%
% LEBESGUE evaluates the density function relating the Lebesgue
% measure to itself (per the Radon-Nikodym theorem) at a given
% set of points. This density function is identically one, so the
% output is simply a vector of ones of the same length as 'x'.
% *****************

m=ones(size(x,1),1);
```

Commuter Generation

This section contains code for generating the commuter functions that we have encountered. In particular, f_{E_9} and $f_{E_{10}}$ were constructed with this code.

tent_commuter.m

This program is used for constructing a commuter between two symmetric tent maps, such as f_{E_9}. It is designed to evaluate the commuter at a specified set of points.

```
function f=tent_commuter(X)

% *****************
% tent_commuter.m
% Author: Scott M. LaLonde
% Date created: 10/6/08
% Last modified: 4/14/09
% *****************
%
% TENT_COMMUTER     Evaluates the commuter between two specified
%                   symmetric tent maps on a collection of points
%                   from the interval [0,1].
%
% Usage:            f=tent_commuter(x)
```

```
%
% Input:
%   x              Vector of N data points from the interval
%                  [0,1]. This should be input as an N-by-1
%                  array.
%
% Output:
%   f              Function values associated with points in x.
%                  This is output as an N-by-1 array.
%
% TENT_COMMUTER evaluates the commuter function between two symm-
% etric tent maps at a given set of points. The heights of the two
% maps should be speficied as global variables called 'H1' and
% 'H2'. The functional equation for the commuter is used to iter-
% atively approximate the values of the function at the points in
% 'x'.
% *****************

% Set parameters for two tent maps. These can be changed to
% compute commuter function between two arbitrary symmetric tent
% maps.
global H1 H2;
a=1/2;

% Use input data to generate data set on [0,1] of approximately
% the same density as X.
xlower=min(X)*rand(floor(min(X)*length(X)/(max(X)-min(X))),1);
xupper=(1-max(X))*rand(floor((1-max(X))*length(X)/(max(X)
          -min(X))),1)+max(X);
x=cat(1, X, xlower, xupper);

% Calculate nearest neighbors in x for functions of x that will
% be input to h.
x1=2*H1*x(x<=1/2);
x2=2*H1*(1-x(x>1/2));

% Calculate sequence of f's, which will converge to commuter
% function.
f=x;
f1=zeros(length(x),1);
for i=1:100
    pp=interp1(x, f, 'linear', 'pp');
    f1(x<=1/2)=a/H2*ppval(pp, x1);
    f1(x>1/2)=1-(1-a)/H2*ppval(pp, x2);
    f=f1;
end

% Output only those function values corresponding to the points
% in the original data set.
f=f(1:length(X));
```

skew_commuter.m

This program is used to construct a commuter between a symmetric tent map and a skew tent map, such as $f_{E_{10}}$. The commuter is also evaluated at a specified number of points.

```
function f=skew_commuter(X)

% *****************
% skew_commuter.m
% Author: Scott M. LaLonde
% Date created: 10/6/08
% Last modified: 4/14/09
%
% SKEW_COMMUTER     Evaluates the commuter between a skew tent map
%                   of height 1 and the symmetric tent map of
%                   height 1 on a collection of points from the
%                   interval [0,1].
%
% Usage:            f=skew_commuter(x)
%
% Input:
%   x               Vector of N data points from the interval
%                   [0,1]. This should be input as an N-by-1
%                   array.
%
% Output:
%   f               Function values associated with points in x.
%                   This is output as an N-by-1 array.
%
% TENT_COMMUTER evaluates the commuter function between a skew
% tent map and a symmetric tent map (both of height one) at a
% given set of points. The point at which the peak of the skew
% tent map occurs should be specified as a global variable called
% 'A'. The functional equation for the commuter is used to iter-
% atively approximate the values of the function at the points in
% 'x'.
% *****************

% Set parameter for skew tent map.
global A;
H1=1; H2=1;
```

```
% Use input data to generate data set on [0,1] of approximately
% the same density as X.
xlower=min(X)*rand(floor(min(X)*length(X)/(max(X)-min(X))),1);
xupper=(1-max(X))*rand(floor((1-max(X))*length(X)/(max(X)
          -min(X))),1)+max(X);
x=cat(1, X, xlower, xupper);

% Calculate nearest neighbors in x for functions of x that will
% be input to h.
x1=2*H1*x(x<=1/2);
x2=2*H1*(1-x(x>1/2));

% Calculate sequence of f's, which will converge to commuter
% function.
f=x;
f1=zeros(length(x),1);
for i=1:100
    pp=interp1(x, f, 'linear', 'pp');
    f1(x<=1/2)=A/H2*ppval(pp, x1);
    f1(x>1/2)=1-(1-A)/H2*ppval(pp, x2);
    f=f1;
end

% Output only those function values corresponding to the points
% in the original data set.
f=f(1:length(X));
```

Set Generation

The code in this section is used to generate some of the more interesting benchmark sets, particularly the Cantor sets.

cantorsample.m

This program generates a random sample of a specified size from the middle-third Cantor set.

```
function c=cantorsample(n)

% *****************
% cantorsample.m
% Author: Scott M. LaLonde
% Date created: 8/1/08
% Last modified: 4/14/09
% *****************
%
% CANTORSAMPLE  Generate a random sample of size n, uniformly
%               distributed on the Cantor set.
%
% Usage:        c=cantorsample(n)
%
% Input
%
%   n           Desired size of random sample. This should be a
%               positive scalar.
%
% Output
%
%   c           Random sample of points in Cantor set. This will
%               be an n-by-1 vector.
%
% CANTORSAMPLE generates a random sample of size n taken from a
% uniform distribution on the middle-third Cantor set.
% *****************

c=sum(2*floor(rand(n,10)*2).*(ones(n,1)*((1/3).^(1:10))),2);
```

SVC_endpoints.m

This program generates the endpoints of a finite approximation of the Smith-Volterra-Cantor set. By finite approximation, we mean the set given by taking only a finite intersection of sets in the construction of the SVC. Such a set will consist of a union of very small intervals, and this code generates the endpoints of those intervals. The output can then be used to select random points in the SVC.

```
function pts=SVC_endpoints(n)

% *****************
% SVC_endpoints.m
% Author: Scott M. LaLonde
% Date created: 3/27/09
% Last modified: 4/14/09
% *****************
%
% SVC_ENDPOINTS     Generate endpoints of the intervals present in
%                   the nth step of the construction of the Smith-
%                   Volterra-Cantor set.
```

```
%
% Usage:            pts=SVC_endpoints(n)
%
% Input
%
%   n               Step at which to terminate construction of
%                   SVC.
%
% Output
%
%   pts             Endpoints left at nth step of construction of
%                   the SVC.
%
% SVC_ENDPOINTS generates the endpoints of the nth step of the
% Smith-Volterra-Cantor set ("fat" Cantor set). The SVC is cons-
% tructed by removing intervals of certain size at a countable
% number of steps, and then intersecting the resulting sets. This
% program essentially approximates the set by taking only a finite
% intersection and returning the endpoints.
% *****************

% Initialize vector to hold endpoints.
pts=zeros(2^n, 2);
pts(1,1)=0; pts(1,2)=1;

% Generate endpoints by computing the midpoint of each interval,
% then adding and subtracting half the current gap. Sort and
% repeat.
for i=1:n
    gap=1/2^(2*i);
    pts=sort(pts);
    mid=(pts(:,1)+pts(:,2))/2;
    mid=flipud(mid);
    new=[mid+gap/2 mid-gap/2];
    new(2^(i-1)+1:2^n, :)=zeros(2^n-2^(i-1), 2);
    pts=pts+new;
end

pts=sort(pts);
```

Plot Generation

These programs were used to generate some of the more complex plots seen in this thesis, particularly those shown in Chapter 3.

fat_rad_sequence.m

This program is designed to generate plots of computed measure versus fattening radius for a specified set and sequence of radii. The plots of this type found in Section 3.2 were generated using this code.

```
function measure=fat_rad_sequence(sample, bounds, N, density, r)

% *****************
% fat_rad_sequence.m
% Author: Scott M. LaLonde
% Date created: 3/24/09
% Last modified: 4/10/09
% *****************
%
% FAT_RAD_SEQUENCE  Compute approximate measure of a subset E of
%                   R^n via Monte Carlo integration. This is by
%                   fattening a random sample of E using various
%                   radii. The resulting data is also plotted.
%
% Usage:            measure=fat_rad_sequence(sample, bounds, N,
%                               density, r)
%
% Input:
%
%   sample          Random sample of size N from the set E. This
%                   should be given as an N-by-n array; that is,
%                   the rows of 'sample' correspond to points in
%                   R^n.
%
%   bounds          Upper and lower bounds of an n-dimensional
%                   interval containing E. The lower bounds are
%                   contained in the first column and the corre-
%                   sponding upper bounds are contained in the
%                   second column. This should be an n-by-2 array.
%
%   N               Number of random points to be used for Monte
%                   Carlo integration. This should be a positive
%                   integer.
%
%   density         Handle for function relating the desired
%                   measure to the Lebesgue measure, per the
%                   Radon-Nikodym theorem.
%
%   r               Vector of radii by which the set is to be
%                   fattened. This should be an M-by-1 array of
%                   non-negative scalars, where M is the desired
%                   number of radii.
%
% Output:
%
%   measure         Values of computed measure corresponding to
%                   the fattening radii. This is output as an
%                   M-by-1 array of non-negative scalars.
%
% FAT_RAD_SEQUENCE computes the approximate measure of a subset of
% R^n based on a random sample of the set. It calls 'mc_int.m' for
% each radius in the array 'r'. Refer to the documentation for
% that file for more details. The computed measures are then plot-
% ted agains the fattening radii.
% *****************

% Run mc_int.m for each value of r.
measure=zeros(length(r), 1);
for i=1:length(r)
    measure(i)=mc_int(sample, bounds, N, density, r(i));
end

% Plot results.
plot(r, measure, '.');
xlabel('Radius', 'fontsize', 12);
ylabel('Computed measure', 'fontsize', 12);
set(gca, 'fontsize', 12);
set(gca, 'plotboxaspectratio', [1 1 1]);
```

commuter_sequence.m

This program generates a surface plot of computed measure versus fattening radius and height for a sequence of commuters. In this case the commuters relate the maps g_1 and g_2, where g_2 is the full shift tent map and g_1 is a shorter symmetric tent map. The plots seen in Section 3.4 were generated in this manner.

```
function measure=commuter_sequence(r, param)

% *****************
% commuter_sequence.m
% Author: Scott M. LaLonde
% Date created: 4/7/09
% Last modified: 4/10/09
% *****************
%
% COMMUTER_SEQUENCE  Generates a surface plot computed measure as
%                     a function of radius and a parameter for a
%                     sequence of parameterized commuters.
%
% Usage:              measure=commuter_sequence(r, param)
%
% Input
%
%   r                 Vector of N radii to be used. This should be
%                     an N-by-1 array of non-negative scalars.
%
%   param             Vector of M parameters for the sequence of
%                     commuters. This should be an M-by-1 array of
%                     positive scalars.
%
% Output:
%
%   measure           Array of computed measure values corresp-
%                     onding to the elements of 'r' and 'param'.
%                     This is output as an N-by-M array of non-
%                     negative scalars.
%
% COMMUTER_SEQUENCE generates a sequence of commuter functions
% between tent maps. In each case, the second map is taken to be
% the standard tent map, and the first is a shorter symmetric tent
% map. Because of this, 'param' is simply a vector of heights to
% be used for the first map. For each parameter value,
% 'fat_rad_sequence.m' is used to calculate the computed measure
% for all radii. This data is then used to generate a surface plot.
% *****************

% Define global variables for use with 'tent_commuter.m'.
global H1 H2;
H2=1;

x=rand(1000, 1);
bounds=[0 1];
N=10000;
measure=zeros(length(r), length(param));

% For each parameter value, construct corresponding commuter and
% generate
% measure values for all radii.
for i=1:length(param)
    H1=param(i);
    sample=tent_commuter(x);
    measure(:, i)=fat_rad_sequence(sample, bounds, N,
              @lebesgue, r);
end
```

```
% Create surface plot.
surf(r, param, measure);
xlabel('Radius', 'fontsize', 12);
ylabel('Height of g_1', 'fontsize', 12);
zlabel('Computed measure', 'fontsize', 12);
set(gca, 'fontsize', 12);
```

ball_plot.m

This code is used to generate an image of a fattened set for a given fattening radius r. In particular, a sample is taken from the set $[0, 1/2] \times [0, 1/2]$, and a ball of radius r is placed around each point in the sample. An image of the resulting set is then produced.

```
function x=ball_plot(r)

% ****************
% ball_plot.m
% Author: Scott M. LaLonde
% Date created: 3/16/09
% Last modified: 4/14/09
% ****************
%
% BALL_PLOT     Generates pictures of a fattened set for a given
%               fattening radius.
%
% Usage:        x=ball_plot(r)
%
% Input
%
%   r           Fattening radius. This should be a non-negative
%               scalar.
%
% Output
%
%   x           Random sample that has been used for the fattening
%               process. This is a 1000-by-2 array.
%
% BALL_PLOT generates an image of a set which has been fattened by
% some radius 'r'. In particular, points are taken from the
% interval [0, 1/2] x [0, 1/2], and a ball of radius r is placed
% around each point. This is then plotted, along with the boundary
% of the interval outlined in blue.
% ****************
% Generate sample of points from interval.
x=rand(1000, 2)/2;
hold on;

% Place a red ball of radius 'r' around each point in 'x'.
t=0:0.01:2*pi;
for i=1:1000
    h=patch(sqrt(r)*cos(t)+x(i,1), sqrt(r)*sin(t)+x(i,2), 'r');
    set(h, 'edgealpha', 0);
end
plot(x(:,1), x(:,2), '.g', 'markersize', 4)

% Plot interval.
plot([0 1/2], [0 0]);
plot([0 1/2], [1/2 1/2]);
plot([0 0], [0 1/2]);
plot([1/2 1/2], [0 1/2]);

xlim([0 1]); ylim([0 1]);
set(gca, 'plotboxaspectratio', [1 1 1]);
```